Universidade de Aveiro
2008
Departamento de Electrónica, Telecomunicações e Informática

CATARINA MARIA DOS SANTOS AMARAL MONTEIRO

SISTEMA INTEGRADO PARA RECOLHA DE INFORMAÇÃO DE REDE

Universidade de Aveiro
2008
Departamento de Electrónica, Telecomunicações e Informática

CATARINA MARIA DOS SANTOS AMARAL MONTEIRO

SISTEMA INTEGRADO PARA RECOLHA DE INFORMAÇÃO DE REDE

dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Dr. Paulo Salvador, Professor Auxiliar convidado e do Dr. António Nogueira, Professor Auxiliar, ambos do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri

presidente

Prof. Dr. Armando José Formoso de Pinho
professor associado da Universidade de Aveiro

Prof. Dr. António Manuel Duarte Nogueira
professor auxiliar da Universidade de Aveiro

Prof. Dr. Joel José Puga Coelho Rodrigues
professor auxiliar do Departamento de Informática da Universidade da Beira Interior

Prof. Dr. Paulo Jorge Salvador Serra Ferreira
professor auxiliar convidado da Universidade de Aveiro

agradecimentos

Ao meu orientador, Prof. Doutor Paulo Jorge Salvador Serra Ferreira cujo conhecimento, sugestões e paciência foram fundamentais na elaboração deste trabalho.
Ao meu co-orientador, Prof. Doutor António Manuel Duarte Nogueira, pessoa que me transmitiu o gosto pela temática de Redes desde a minha primeira abordagem ao tema bem como, no auxilio da preparação deste trabalho.
O meu mais profundo agradecimento é dirigido à minha mãe, por todo o amor, apoio e força para caminhar e vencer todas as dificuldades que ultrapassámos sempre juntas.
Ao meu pai e padrinhos, que foram decisivos no meu desenvolvimento pessoal.
Ao meu namorado, por me ter acompanhado ao longo de todo curso e me ter dado sempre muito apoio e muitas palavras de incentivo, assim como também à minha prima Isabel Tavares e ao meu primo Carlos Arede.
Aos meus amigos e colegas adquiridos ao longo do curso, por estarem sempre ao meu lado dispostos a ouvir-me e apoiar-me.
Esta dissertação foi elaborada também com o apoio do Instituto de Telecomunicações, que disponibilizou os equipamentos necessários no desenvolvimento do trabalho.

palavras-chave

rede, recolha, monitorização, informação, serviços, equipamentos de rede, segurança, interface web.

resumo

Devido à enorme expansão das redes de telecomunicações, ao rápido crescimento da Internet, do número de serviços e da heterogeneidade das tecnologias actuais, torna-se impossível e muito caro a gestão manual da rede e sua análise forense. Necessita-se assim, que existam sistemas que façam uma constante monitorização da rede de forma automática.
Estes sistemas servirão para detectar qualquer tipo de problema na rede, inclusive os de segurança, algo que está no topo das preocupações de qualquer organização com recursos informáticos e que deve ser encarado como um processo contínuo de optimização dos sistemas e das políticas de cada empresa, um processo que não pára no tempo
Esta dissertação aborda um sistema capaz de realizar, de um modo autónomo, tarefas de recolha, processamento e armazenagem numa base de dados, das informações importantes da rede, nomeadamente registos de ficheiros de log de servidores e tráfego nos equipamentos de rede. Através de uma interface Web, também desenvolvida no âmbito deste trabalho, é possível ao administrador ter uma visão clara e unificada dos recursos recolhidos.

keywords

network, gathering, monitorization, information, services, networking equipment, security, web interface.

abstract

Due to enormous expansion of the telecommunication network, the fast growth of the Internet, the number of services and the heterogeneity of present technologies, it becomes impossible and very expensive to manually manage the network and its forensic analysis. It is necessary that systems which are constantly and automatically monitoring the network exist.
These systems will serve to detect any type of problem in the network, also of security. Security is one of the top concerns of any organization with computing resources and it's fundamental to think that this is a continuous process of optimization of the systems and the policies of each company, a process that does not stop over time.
This dissertation addresses a system capable of performing, in an autonomous way, gathering tasks, processing and storage in a database, of important network information, including records of log files from servers and traffic in the network equipment. Through a Web interface, also developed in the scope of this dissertation, it is possible for an administrator to have a clear and unified vision of the collected resources.

Índice de Conteúdo

Índice de Figuras

Índice de Tabelas

1 Introdução

Devido à enorme expansão das redes de telecomunicações, ao rápido crescimento da Internet, do número de serviços e da heterogeneidade das tecnologias actuais, torna-se impossível e muito caro a gestão manual da rede. O número de ameaças à segurança das redes e seus utilizadores é muito maior, surgindo a necessidade extrema de haver uma constante monitorização da rede. Há assim, a necessidade de existir sistemas que façam uma gestão e monitorização de redes de forma automática, permitindo o planeamento eficiente, um melhor serviço e um maior conhecimento da rede.

1.1 Modelos de gestão

A International Organization for Standardization (ISO) definiu cinco áreas conceptuais na gestão de redes. Isolando os problemas de gestão em áreas distintas, este modelo permite conceptualizar soluções que são optimizadas para os problemas específicos de cada área funcional. Estas áreas resumem-se à sigla FCAPS:

- **F**ault Management (gestão de falhas): detecção, isolamento e se possível a correcção de comportamentos anormais;
- **C**onfiguration Management (gestão de configuração): monitorização e configuração para gestão de várias versões de hardware e software;
- **A**ccounting Management (gestão de contabilidade): medição de parâmetros como a utilização da rede, determinar custos e outros, de maneira a regular e minimizar problemas da rede;
- **P**erformance Management (gestão de desempenho): avaliação e medição de desempenho da rede;
- **S**ecurity Management (gestão de segurança): controlar o acesso seguro aos recursos da rede.

Este trabalho tem contexto na área Security Management.

1.2 Monitorização

A monitorização é o procedimento que permite acompanhar e controlar o processo de intervenção e identificar eventuais desvios face ao que foi previsto num momento inicial. A monitorização deve assentar num sistema de registo contínuo de dados e de acções, visando acompanhar e controlar de forma continuada todos os processos existentes na rede.

Hoje em dia há inúmeras aplicações disponíveis que monitorizam diversos aspectos de uma rede de telecomunicações. Exemplos são: Ethereal, Wireshark, Snort, Netcat, TCPDump, AirSnort, Ntop, Snoop. Uma parte deste trabalho consistirá em monitorizar a rede através dos ficheiros de log registados nos seus servidores. Porém, é de salientar que existem no mercado empresas que se dedicam ao desenvolvimento deste tipo de software. Um dos programas é o Lire, que suporta alguns serviços: *Web* (Apache, Boa, iPlanet), FTP (BSD ftpd, ProFTPD, Wu-Ftpd), DNS (Bind8, Bind9), e-mail (ArGoSoftMail. Exim, Postfix, Qmail, Sendmail) entre outros [Lire2008]. Outro programa é o XpoLog que faz dentre outras coisas a análise estatística dos logs [Xpolog2008].

1.3 Motivações da dissertação

De maneira a efectivamente gerir os servidores de uma rede é necessário haver *feedback* sobre as actividades e desempenhos dos servidores em causa. Com este intuito há a criação e armazenamento constante de ficheiros de log no sistema de ficheiros dos servidores. Estes ficheiros de log contêm uma grande quantidade de informação e o tipo de informação varia consoante o serviço em estudo. Uma vez que ocupam muito espaço em disco, é recomendado que depois da sua análise estes sejam apagados do sistema.

A análise destes ficheiros é uma técnica que permite determinar *a posteriori* violações de segurança. Em ataques passivos, o intruso escuta às escondidas e não modifica qualquer tráfego da rede ou ficheiro num sistema operativo porém, em ataques activos, o intruso pode transmitir, apagar e modificar mensagens, alterar ficheiros. Deste modo, é recomendado que apenas o administrador tenha direito de acesso aos ficheiros de log, principalmente do direito de escrita, pois maliciosos clientes poderiam altera-los e tornar a detecção das violações de segurança impossíveis por este meio.

Este trabalho, em parte consiste na recolha e análise destes ficheiros de log de forma a que as informações sejam apresentadas de forma clara e compreensível.

Para além disto, de maneira a monitorizar de uma forma mais abrangente uma rede é importante saber o tráfego que atravessa os equipamentos da rede, interfaces utilizadas, endereços IP e MACs ligados às interfaces, portos ocupados, entre outros.

Assim, outra parte deste trabalho recai na recolha e análise de todas as informações relevantes dos equipamentos que possam auxiliar na detecção de actividades maliciosas, através do protocolo de gestão de redes Simple Network Management Protocol (SNMP).

1.4 Estrutura da dissertação

A dissertação encontra-se organizada em 8 capítulos. Nestes serão abordados os seguintes aspectos:

- Capítulo 1 – *Introdução*.
- No capítulo 2 – *Estado da arte* – descreve-se a função e o porquê de serem, cada vez mais, necessários sistemas de monitorização em qualquer rede de comunicações. São apresentados também alguns dos softwares mais utilizados na monitorização de redes.
- No capítulo 3 – *Arquitectura do sistema* – descreve-se o sistema criado neste trabalho e a responsabilidade de cada bloco que o compõe.
- O capítulo 4 – *Servidores* – é dedicado a descrever os serviços estudados, as suas configurações, ficheiros de log, e scripts para processamento destes ficheiros.
- No capítulo 5 – *Equipamentos de rede* – descreve-se em que consiste o protocolo SNMP. Seguidamente apresentam-se todos os dados que foram lidos dos equipamentos através deste protocolo.
- No capítulo 6 – *Base de dados* – é apresentado o servidor de base de dados utilizado neste trabalho, MySQL, os seus tipos de campos, comandos e restrições. Também se descrevem as bases de dados criadas e as suas tabelas, para armazenar os dados obtidos na monitorização da rede (de servidores e equipamentos de rede).
- No capítulo 7 – *Interface Web* – é apresentada a interface *Web* desenvolvida. Através desta, o administrador da rede pode adicionar e remover serviços e equipamentos do sistema de monitorização e visualizar de forma clara os dados recolhidos e armazenados. Também é apresentada a linguagem utilizada na construção desta interface, o PHP, seus comandos e como é feita a interligação com o servidor MySQL.
- No capítulo 8 – *Conclusões* – são apresentadas as principais conclusões do trabalho e sugestões para desenvolvimento futuro.

1.5 Contribuições

Através de shell scripts fez-se:

- Desenvolvimento de um sistema de recolha de toda a informação dispersa nos servidores e equipamentos numa rede.
- Análise dos ficheiros de log dos principais serviços utilizados hoje em dia no mundo das redes.
- Análise dos dados de equipamentos de rede: routers e switches.

- Armazenamento de toda a informação em base de dados MySQL.

Através de scripts PHP fez-se:

- Desenvolvimento de uma interface *Web* que proporciona ao administrador uma visão clara das ocorrências e actividades na rede, podendo assim correlacionar todos os dados que sejam importantes para o seu trabalho.

1.6 Notações utilizadas

Sempre que, ao longo deste trabalho, a tradução não exprima o significado original serão utilizados estrangeirismos, os quais são escritos em itálico.

Ao longo da dissertação encontram-se acrónimos e siglas cujo significado é apresentado aquando a sua primeira ocorrência e também no fim do trabalho, secção Acrónimos e Siglas.

As referências bibliográficas utilizadas neste documento são evocadas entre parêntesis recto e apresentadas na secção Referências.

2 Estado da arte

2.1 Introdução

A monitorização da rede torna-se cada vez mais um processo necessário por parte de instituições e seus administradores de rede. Esta necessidade deve-se principalmente a três factores: em primeiro lugar, o tamanho das redes e o seu tráfego continua a aumentar e cada vez mais dispositivos de rede precisam ser eficientemente geridos, exigindo uma melhor escalabilidade nos designs de gestão das redes. Em segundo, redes com diferentes tecnologias, heterogéneas, devem co-existir e interagirem entre si. Por fim, há uma grande preocupação também económica por trás disto, deste modo, a intervenção humana deve apenas estar presente no mais alto nível de abstracção e generalidade.

Uma boa monitorização da rede é necessária também para garantir a sua segurança. O conceito de segurança está definitivamente instalado no centro das atenções de todos os administradores de redes. As análises realizadas por diversas entidades à segurança da Internet apontam num único sentido: as ameaças são mais cada vez maiores e acontecem com mais assiduidade. A segurança é uma preocupação fundamental para qualquer organização com recursos informáticos e, sobretudo, com ligações à Internet ou com redes privadas nacionais e internacionais.

Uma barreira psicológica que hoje em dia ainda se levanta é o pensar que por uma empresa ser demasiado pequena ou pouco importante, não é um alvo de ataques. Com a difusão de informação sobre bugs ou falhas nos sistemas, muitos hackers atacam aleatoriamente endereços na Internet, que correspondam a sistemas potencialmente vulneráveis, com o intuito de provocarem danos sempre que encontram um alvo fácil. Muito raramente o ataque visa um alvo específico e conscientemente escolhido. Na propagação de vírus e worms também não há preferências, provocam estragos indiscriminadamente e na maioria das vezes são difundidos por e-mail. Assim, este sentimento de que "não me afecta a mim", deve ser abandonado, para que se possa encarar seriamente o tema segurança.

É fundamental pensar-se na segurança como um processo contínuo de optimização dos sistemas e das políticas de cada empresa, um processo que não pára no tempo e cujo objectivo é minimizar, a cada dia, o risco inerente a uma organização que depende de sistemas informáticos e de telecomunicações. Não pode ser encarada como um projecto localizado no tempo ou um conjunto de sistemas criados e instalados para combater vírus, hackers e outros fenómenos semelhantes que acarretam prejuízos para uma empresa sempre que se verificam.

Devem ser, pelos motivos referidos, implementados mecanismos de monitorização que de forma autónoma façam a recolha de informações dispersas em toda a rede, identifiquem eventuais anomalias e que apresentem ao administrador apenas o que é relevante.

O processo de monitorização consiste em verificar computadores, equipamentos, serviços, sistemas que constituam a rede e permitam que um administrador de redes defenda a rede que está sob a sua administração e eventualmente aperfeiçoe a mesma. A monitorização de sistemas e de toda a infra-estrutura, permite que não só sejam mantidas estatísticas dos serviços, mas também que sejam rapidamente identificados problemas e partir imediatamente para a resolução dos mesmos.

Monitorizar redes à procura de vulnerabilidades, de falhas e manter uma política de segurança são tarefas que não podem ser negligenciadas. Tem havido uma preocupação constante por parte de entidades e empresas no desenvolvimento de *software* de monitorização. Nos seguintes sub-capítulos apresenta-se uma lista de alguns disponíveis no mercado. De entre eles encontram-se: monitorização de base de dados, monitorização de ficheiros de log, monitorização da rede e sistemas, monitorização de tráfego, análise de protocolos e captura de pacotes, monitorização de segurança, monitorização SNMP, monitorização *Web*.

2.2 AlertBot

AlertBot [Alertbot2008] é um software de monitorização de páginas *Web* de empresas e servidores Internet. Os dados são representados graficamente permitindo que a empresa tome correctas decisões. Quando páginas *Web* ou servidores falham, há imediatamente o diagnóstico do problema e o envio de um alerta aos gestores de rede da empresa por e-mail ou SMS. O serviço disponibilizado serve tanto para pequenas, quanto para grandes empresas.

2.3 Activeworx

Activeworx [Crosstecsecurity2008] é um sistema de gestão de informações de segurança que proporciona monitorização de logs, com sistema de alerta, auditoria e relatórios sobre análise forense. Activeworx consolida a informação e gera alarmes. A recolha é feita em diferentes equipamentos na rede e há diferentes tipos de base de dados que armazenam dados retirados dos seguintes eventos: sistema de detecção de intrusos, incluindo Snort; Syslog; Firewall; Scans; pacotes TCPDump; logs.

2.4 GFI

GFI Network Server Monitor [GFI2008] monitoriza a rede e servidores de falhas. Os alertas podem ser enviados por e-mail, pager, SMS. Acções como reiniciar um computador, um serviço ou correr um script podem ser feitos automaticamente. As regras internas de monitorização incluem: Exchange Server 2000/2003, MS SQL, base de dados Oracle e ODBC, utilização CPU,

servidores FTP & HTTP, Active Directory & NTDS, espaço do disco, Event Log, TCP, UDP, ICMP/Ping, servidores SMTP & POP3 Mail, impressoras, UNIX Shell Scripts, SNMP & Terminal Server.

2.5 ActiveXperts

ActiveXperts [ActiveXperts2008] é um dos líderes em monitorização de vários aspectos de uma LAN e WAN. Faz a gestão de servidores, impressoras, equipamentos de rede, base de dados, entre outros. Detecta, corrige problemas e apresenta bastante escalabilidade. É capaz de processar mais do que trinta e dois VBScripts simultaneamente.

Quando um problema é detectado, o administrador ou responsável pela rede é imediatamente notificado por mensagem na rede, e-mail, pager ou SMS. O produto inclui construir regras na monitorização de: Active Directory, base de dados ADO, ADSI, Disk Drives, utilização CPU, espaço em disco, Event Log, Exchange 2000, FTP sites, HTTP, ICMP/Ping, MS SQL Databases, Novell NDS, servidores NNTP, NTDS, servidores NTP, base de dados ODBC, base de dados Oracle (SQLNet), POP3 Mail servers, impressoras, servidores de e-mail SMTP, SNMP, portas TCP, UDP, UNIX Shell Scripts, VBScript, WMI, e outros.

2.6 HealthMonitor

HealthMonitor [HealthMonitor2008] é uma das ferramentas mais detalhadas da administração de sistemas disponíveis. Controla-se através do **HealthMonitor** todos os utilizadores e estações de trabalho em uma rede.

Inclui cinquenta verificações diferentes para o sistema e a rede, entre elas: conectividade, base de dados, e-mail, desempenho, SQL Server, Exchange. Dinâmica de acções, envio de e-mail, SMS, quando algum erro for detectado. Descoberta de hardware e periféricos.

Através de uma interface *Web* bastante *userfriendly*, pode-se ver e analisar as informações do sistema, gerir clientes, verificar o desempenho, corrigir problemas, reiniciar serviços, correr scripts, gerar relatórios entre outros.

2.7 Nagios

O **Nagios** [Nagios2008] é uma ferramenta *open-source,* compatível com sistemas operativos UNIX, para monitorização de rede com interface *Web*. Esta ferramenta tem a vantagem da possibilidade de adição de novas funcionalidades através de *plugins*. Assim, este é constituído por um módulo central e por *plugins* escritos em C, Perl e Shell scripts. O programa central invoca os *plugins* para efectuarem a monitorização. Das funcionalidades destacam-se a monitorização de serviços de rede (SMTP, http, NNTP, SSH), monitorização de recursos de hosts, sistema de alarme

quando ocorre algum problema, interface *Web* para visualização da rede, notificações e histórico de problemas.

2.8 HP OpenView

O **HP OpenView** [HPOpenView2008] é actualmente uma das plataformas de gestão mais divulgadas e utilizadas. Esta realidade prende-se com o facto de ser uma plataforma de gestão com uma interface muito simples e completa, não sendo também alheia a existência de módulos adicionais, desenvolvidos por outros fabricantes de equipamentos de rede.

Apresenta várias ferramentas de gestão e monitorização fundamentais em IT organizações. De acordo com a International Data Corporation, HP é o dono de mais soluções. Esta tecnologia obtém automaticamente da rede informações que podem ser utilizadas para prever falhas, antes que estas possam afectar o serviço, e permite ainda a detecção mais rápida e exacta do problema na raiz. As soluções fornecem a gerência detalhada do evento, a monitorização de desempenho dinâmica, análise, a alerta automatizada, o relatório, virtualização, representação gráfica. Permitem também automatização da reconfiguração de um data center, permitindo que os directores e responsáveis de informática não se limitem a uma gestão manual e reactiva sem acrescentarem novo hardware ou novas aplicações. Duas de suas soluções são a HP OpenView Automation Manager e o HP OpenView Service Desk.

2.9 Síntese

Neste capítulo foi descrito o quanto a monitorização de redes se tornou importante na temática redes de telecomunicações. Foram apresentados também alguns exemplos de software deste tipo de desenvolvimento. Outros podem ser citados, tais como: AspEmail, ManageEngine, Netmon, SQLCentric, OpManager, MonitorMagic, ipMonitor, Cuevision, OpManager.

3 Arquitectura do sistema

3.1 Introdução

O trabalho desenvolvido é um sistema de monitorização de redes. Pretendeu-se desenvolver uma ferramenta capaz de realizar, de um modo autónomo, tarefas de recolha e processamento de dados importantes da rede, tais como ficheiros de log de servidores e dados dos equipamentos.

A arquitectura deste trabalho consiste na:

- Recolha de informação e uso de uma abordagem *manager of managements*: tecnologia cliente/servidor, protocolo SNMP.
- Filtragem, processamento e armazenamento dos dados em uma base de dados. Processos realizados na máquina do administrador.
- Visão unificada dos recursos: através de uma interface *Web*.

Na figura abaixo apresenta-se a arquitectura desenvolvida:

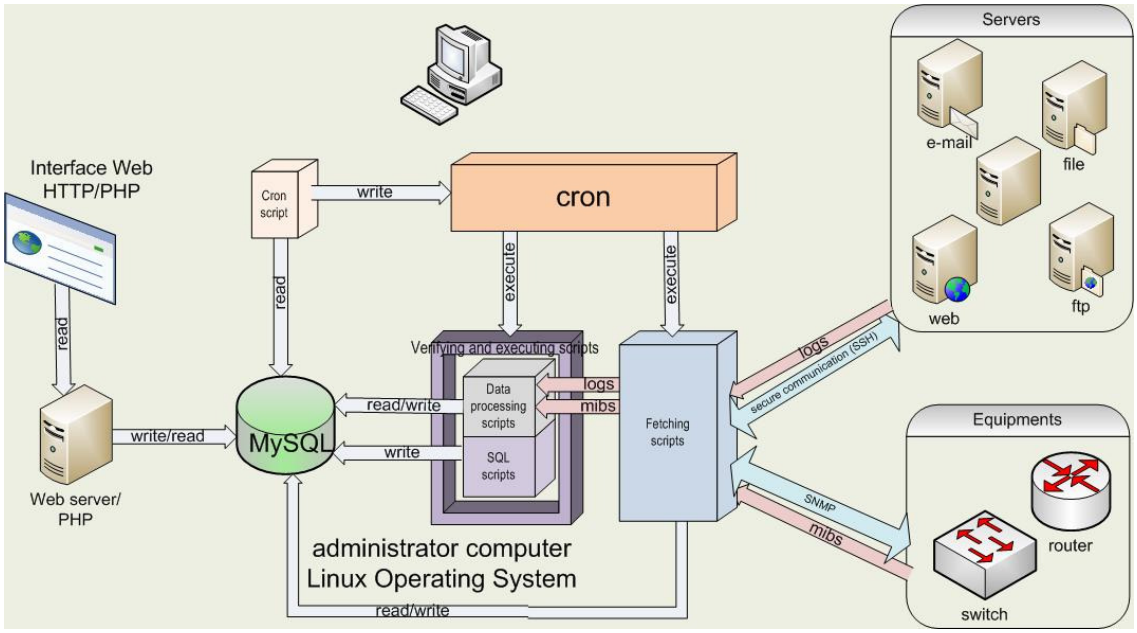


Figura 1 - Arquitectura do sistema

Para viabilizar este sistema foram necessárias tomar algumas decisões. A primeira, e mais determinante, foi admitir que o administrador utiliza um computador com sistema operativo Linux. É necessária pois, utilizou-se o CRON para agendamento dos restantes processos do sistema.

Outro motivo deve-se ao facto de utilizar shell scripts na base de todo este trabalho. Os shell scripts foram utilizados pois apresentam inúmeras vantagens:

- É executado nativamente.
- É uma linguagem extremamente poderosa, já que une *bash* à vasta gama de utilitários presentes em um sistema UNIX.
- Linguagem não é compilada, e sim interpretada.
- Bastante rápida.
- Tratamento em sistemas complexos.
- Sintaxe simples.
- Facilidade no processamento de dados.
- Permite com o auxílio do CRON a autonomia total das tarefas.

Outra das opções foi utilizar o servidor de base de dados MySQL. A opção primeiramente foi tomada por ser um software livre e consequentemente, pode ser utilizado num trabalho académico. Outros recursos são: a capacidade de manipular muitos registos, excelente desempenho, execução rápida de comandos, facilidade de uso, pode ser multi-tarefa e multi-usuário. Além de todos estes factores, é compatível com o PHP, linguagem de desenvolvimento da interface *Web* utilizada neste desenvolvimento. O PHP também apresenta licença gratuita, uma óptima eficiência e velocidade de processamento.

Todo o código dos shell scripts, scripts SQL e PHP é apresentado na íntegra no CD-ROM entregue com a dissertação segundo as normas aprovadas.

Nos sub-capítulos seguintes segue-se a descrição de cada uma das partes deste sistema.

3.2 CRON

Este é um serviço disponível em sistemas operativos Linux e outros Unix. É possível através deste agendar processos no sistema. Neste trabalho recorreu-se ao CRON para a execução dos fetching scripts, ver sub-capítulo 3.3, e dos verifying and executing scripts, ver sub-capítulo 3.4.

Utiliza-se um ficheiro de texto, o crontab, com a lista dos comandos para execução e os seus respectivos tempos. O sistema operativo mantém um crontab para cada utilizador do sistema. Neste sistema utiliza-se o crontab do próprio administrador. Para a visualização do crontab o administrador deve como administrador digitar o comando **crontab -l**.

Para editar o crontab foi utilizado o shell script **script_tempos_cron.sh** com comandos que seguem o seguinte formato:

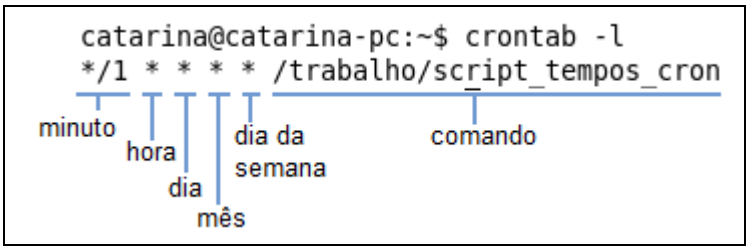


Figura 2 - Formato comandos CRON

- minuto: valores 0 a 59.
- hora: valores 0 a 23.
- dia: valores 1 a 31.
- mês: valores 1 a 12.
- dia da semana: valores 0 a 6 (domingo = 0).
- comando: string com o caminho completo do comando (script) que será executado pelo shell.

Na definição dos parâmetros anteriores quando se tem:

- *: significa que é para todo.
- */x: significa que o intervalo entre execuções é de x.
- x,y: significa que é executado no instante x e no instante y. Leia-se instante como minuto, hora, dia, mês ou dia da semana.

No exemplo da figura 2, temos que o **/trabalho/script_tempos_cron** será executado de um em um minuto, em todas as horas, todos os dias do mês, em todos os meses, em todos os dias da semana.

No âmbito deste trabalho apenas se variou o intervalo de tempo em minutos entre cada execução de cada fetching script e verifying and executing scripts. Os outros parâmetros são definidos com *. Este intervalo de tempo em minutos é definido pelo administrador na interface *Web* e, através da interligação php/mysql, escritos na base de dados. O script **script_tempos_cron.sh** lê da base de dados, através do comando SELECT (descrito no Anexo VI), os tempos e os identificadores e nomes dos servidores ou equipamentos, dependendo de qual dos dois se tratar, e escreve no crontab em todos os minutos actualizando-o com os novos valores da base de dados.

3.3 Fetching Scripts

Os fetching scripts são os scripts que irão recolher remotamente as informações de monitorização estudadas neste trabalho.

3.3.1 Log Fetching Scripts

Há segurança na busca dos ficheiros de log remotamente tanto para servidores em máquina Linux quanto em Windows pois, para isto, utilizaram-se servidores SSH nas máquinas servidor. Nos servidores em Linux utilizou-se a versão OpenSSH_4.3 e em Windows a versão OpenSSH for Windows v3.8.1.

Para a busca dos ficheiros de log dos servidores utiliza-se o shell script **script_get_log.sh** com parâmetro de entrada o identificador do servidor. Este script faz a leitura, através do comando SELECT, dos dados presentes na base de dados do servidor (especificado através do parâmetro de entrada) necessários para esta busca: IP *address*, *username*, *password*, caminho do log no servidor, caminho que irá ser armazenado o log na máquina do administrador e qual o sistema operativo do servidor.

De seguida este script invoca e passa os dados citados anteriormente como parâmetros à outro script, o **script_scp.sh**. O **script_scp.sh** utiliza a ferramenta **Expect** [Expect2008], disponível nos repositórios de aplicações para Linux. É este último que efectivamente recolhe os logs dos servidores para o computador do administrador através do comando SCP cujo formato se apresenta na figura seguinte:

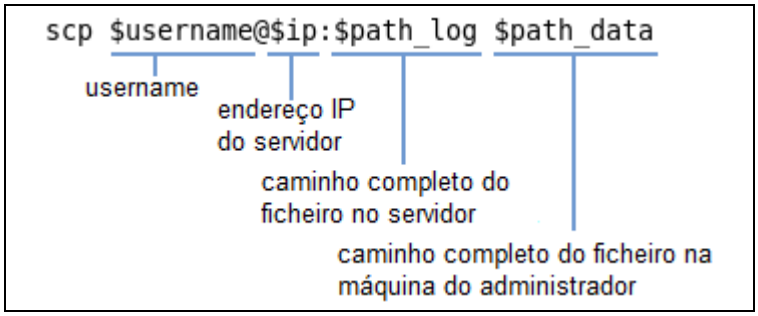


Figura 3 - Comando SCP

A opção de utilizar o script **script_scp.sh** prende-se ao facto de que o comando **SCP** requisita ao utilizador a *password* do servidor e aguarda que o utilizador a escreva no terminal. Como todo o sistema é constituído por scripts que são executados automaticamente pelo CRON e consequentemente sem a acção de nenhum utilizador, era impossível a passagem da *password*. Assim, através da ferramenta **Expect** que é uma ferramenta de automatização de aplicações interactivas, é passada a *password*. Também não era possível utilizar apenas este script do tipo EXPECT pois este não permite a leitura dos dados que necessitamos da base de dados.

No código do **script_scp.sh** também é possível observar o uso do comando **sleep 45**. O que o **sleep** faz é esperar um determinado tempo em segundos, 45 segundos neste caso, para passar a *password*. É necessária esta espera, ou tempos superiores, pois como o acesso aos servidores é

remoto, a conexão muitas vezes pode ser demorada e sem este comando o que aconteceria é que o script passaria a *password* antes desta ser requisitada.

Depois do processo de recolha dos logs terminado, o **script_get_log.sh** actualiza na base de dados, através do comando UPDATE (descrito no Anexo VI), a hora e o dia em que houve a busca dos logs.

No **script_get_log.sh** também há a conversão de formatos dos ficheiros entre sistemas Unix/Windows. Esta operação é necessária quando os ficheiros de log são provenientes de máquinas Windows, pois em sistemas Unix não existe o CR (carriage return). Assim, um ficheiro que é gerado no Windows numa máquina Linux (máquina do administrador) fica como na figura a seguir:

```
abcde^M
abcde^M
abcde^M
abcde^M
```

Figura 4 - Formato ficheiro Windows num sistema Unix

Para solucionar o problema recorreu-se ao pacote **tofrodos** [DOStoUNIX2007], disponível nos repositórios de aplicações para Linux. Também poderia ser feito com o comando **sed**. No conteúdo da instalação temos os binários unix2dos e dos2unix. Utiliza-se neste trabalho apenas o dos2unix e a sintaxe é:

```
dos2unix ficheiro_de_log
```

Figura 5 - Formato comando dos2unix

3.3.2 SNMP Fetching Scripts

O script **script_snmp.sh** é o responsável por recolher nos equipamentos de rede as informações requisitadas pelo administrador. Apresenta como parâmetro de entrada o nome do equipamento. Este também necessita de ler dados do equipamento em questão contidos na base de dados: a *community string* e o *IP address* do equipamento, e se se trata de um router ou um switch. Efectua estas leituras através do comando SELECT.

Para a recolha da informação dos equipamentos de rede através de linha de comandos utiliza-se na máquina Linux do administrador o pacote **snmp**, disponível nos repositórios de aplicações para Linux, que instala as aplicações do **Net-SNMP**. Versão utilizada é a 5.2.3.

3.3.2.1 Net-SNMP

Começou a ser desenvolvido em 1992 na Universidade Carnegie-Mellon e é um conjunto de aplicações usado para implementar SNMPv1, SNMPv2 e SNMPv3 usando IPv4 e IPv6 [NetSNMP2008]. Existem versões para diversos sistemas operativos, inclusive Windows. Este

fornece a *framework* para a troca de informações de gestão entre clientes e agentes. Os pedidos para os agentes são feitos através de linhas de comando, possibilitando o uso de shell scripts. Os comandos utilizam o seguinte formato:

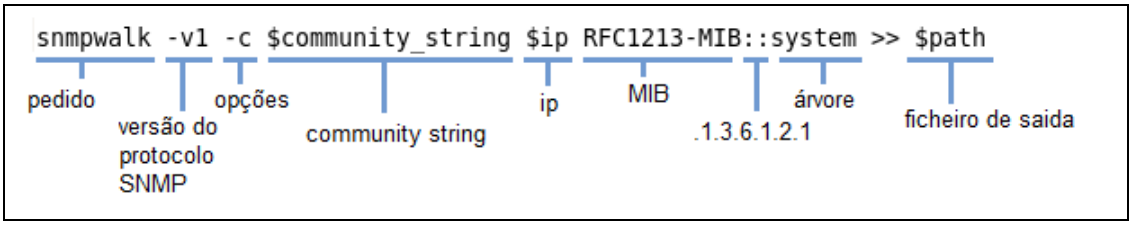


Figura 6 - Comando SNMP

Campos:

- Pedido: tipo de pedido SNMP. Pode ser snmpget, snmpgetnext, snmptable, snmpwalk, snmpinform, entre outros. O snmpwalk foi o utilizado neste trabalho pois obtêm todas as informações da árvore especificada através de sucessivos comandos snmp getnext. Estes pedidos são melhor explicados no capítulo Monitorização dos equipamentos de rede.
- Versão do protocolo SNMP: pode ser v1, v2c, v3. Optou-se apenas por utilizar o SNMPv1 (motivo explicado no capítulo Monitorização dos equipamentos de rede)
- Opções: -c, -d, -m, entre outras (ver Anexo V). A única opção utilizada foi a -c, que permite especificar a community string.
- Community string: valor da *community string* do equipamento, esta varia se for *read-only*, *read-write*.
- IP: Endereço IP do equipamento na rede.
- MIB: MIB utilizada, que apresenta os dados desejados pelo administrador e que o equipamento suporte. Neste trabalho apenas analisamos a RFC1213-MIB e IF-MIB.
- .1.3.6.1.2.1: representado pelos "::", indica .iso.org.dod.internet.mgmt.mib-2.
- Árvore: último campo da árvore. Através do conjunto do campo anterior com este obtêm-se todas as informações referentes à sub-árvore iso(1).identified-organization(3).dod(6).internet(1) management(2).mib-2(1).system(1).
- Ficheiro de saída: o resultado deste comando é encaminhado para um ficheiro do tipo texto (.txt).

Na figura seguinte observa-se a saída do comando explicado anteriormente aplicado a um switch na rede:

```
SNMPv2-MIB::sysDescr.0 = STRING: Cisco IOS Software, C3750ME Software (C3750ME-I5-M), Version 12.2(25)SEG1, RELEASE SOFTWARE (fc1)
Copyright (c) 1986-2006 by Cisco Systems, Inc.
Compiled Mon 07-Aug-06 20:26 by myl
SNMPv2-MIB::sysObjectID.0 = OID: SNMPv2-SMI::enterprises.9.1.574
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (61997) 0:10:19.97
SNMPv2-MIB::sysContact.0 = STRING: pmoreira@av.it.pt
SNMPv2-MIB::sysName.0 = STRING: Switch4
SNMPv2-MIB::sysLocation.0 = STRING: IT Network Lab. 1
SNMPv2-MIB::sysServices.0 = INTEGER: 6
SNMPv2-MIB::sysORLastChange.0 = Timeticks: (0) 0:00:00.00
```

Figura 7 - Saída comando SNMP

3.3.2.2 iReasoning MIB browser

É uma ferramenta de gestão SNMP. Suporta todas as versões SNMP (v1, v2c e v3), Ipv6, trap sender e trap receiver, ferramentas ping e traceroute, vista por tabela nas MIB tables. Pode ser utilizada em sistemas Windows, Mac OS X, Linux e outros sistemas UNIX. É escrito em Java, assim sendo um pré-requisito é ter instalado no computador o Java. Utilizadores podem carregar MIBS standard ou proprietárias, porém só se podem carregar cinco MIBs simultaneamente.

Utilizou-se esta ferramenta apenas para se ter acesso à descrição dos dados SNMP analisados neste trabalho.

3.3.2.3 Router & Switch Fetching Scripts

Neste sub-capítulo descreve-se como foi efectuada a recolha dos dados dos equipamentos, porém, detalhes do protocolo utilizado para isto, SNMP, serão descritos no capítulo Monitorização dos equipamentos de rede.

Para monitorização destes equipamentos, através do protocolo SNMP, utilizou-se um router modelo Cisco 3825 e três switches Cisco modelo Catalyst 3750 Metro. Para a activação do protocolo SNMP configuraram-se os equipamentos com os seguintes comandos:

Router	Switch
Enable	enable
configure terminal	configure terminal
snmp-server community public RO	set snmp community read-only public
snmp-server community private RW	set snmp community read-write private
exit	exit
write memory	write memory

Tabela 1 - Comandos activação SNMP

Os comandos da figura anterior configuram as *community string* dos equipamentos. Consideraram-se aqui os valores padrão mas, poderiam ser configurados outros valores. Para permissão de leitura utiliza-se a *community* **public** e de escrita a *community* **private**.

Através do script **script_snmp.sh**, fez-se a recolha, em duas partes, dos dados que fazem parte de um dos principais grupos de gestão: o MIB-II com OID 1.3.6.1.2.1 ou **iso.org.dod.internet.mgmt.mib-2**. A primeira parte retirou-se da RFC1213-MIB e a segunda da IF-MIB. A árvore **iso.org.dod.internet.mgmt.mib-2** encontra-se na figura seguinte, e o conteúdo retirado das MIBS apresenta-se nas tabelas seguintes, 2 e 3 respectivamente:

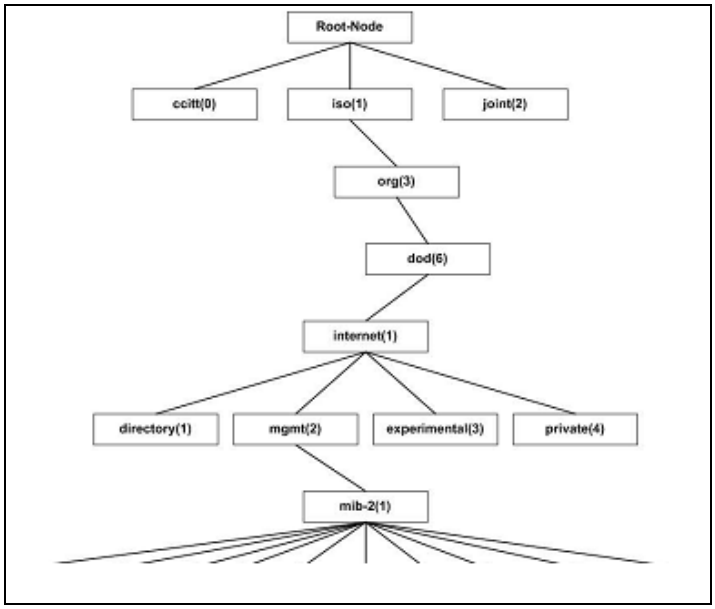


Figura 8 - Árvore MIB-II

Nome da árvore iso.org. internet.mgmt. mib-2.(x)	**OID**	**Campos lidos**	**Descrição**
System	1.3.6.1.2.1.1	sysDescr	Define uma lista de objectos que pertencem ao sistema operacional e dados do equipamento gerido.
		sysObject	
		sysUpTime	
		sysContact	
		sysName	
		sysLocation	
		sysServices	
Interfaces	1.3.6.1.2.1.2	ifNumber	Mantém o histórico do estado e características das interfaces do equipamento gerido.
		ifIndex	
		ifDescr	
		ifType	

		ifMtu	
		ifSpeed	
		ifPhysAddress	
		ifAdminStatus	
		ifOperStatus	
		ifLastChange	
		ifInOctets	
		ifInUcastPkts	
		ifnNUcastPkts	
		ifInDiscards	
		IfInErrors	
		ifInUnknownProtos	
		ifOutOctets	
		ifOutUcastPkts	
		ifOutNUcastPkts	
		ifOutDiscards	
		ifOutErrors	
		ifOutQLen	
at	1.3.6.1.2.1.3	atifIndex	Endereço ip e MAC das entidades ligadas as interfaces do equipamento gerido.
		atPhysAddress	
		atNetAddress	
ip	1.3.6.1.2.1.4	ipForwarding	Mantém histórico de diversos aspectos relevantes do protocolo IP, inclusive roteamento.
		ipDefaultTTL	
		ipInReceives	
		ipInHdrErrors	
		ipInAddrErrors	
		ipForwDatagrams	
		ipInUnknownProtos	
		ipInDiscards	
		ipInDelivers	
		ipOutRequests	
		ipOutDiscards	
		ipOutNoRoutes	
		ipReasmTimeout	
		ipReasmReqds	
		ipReasmOKs	
		ipReasmFails	
		ipFragOKs	
		ipFragFails	
		ipFragCreates	
		ipAdEntAddr	
		ipAdEntIfIndex	
		ipAdEntNetMask	
		ipAdEntBcastAddr	
		ipAdEntReasmMaxSize	

		ipRouteDest	
		ipRouteIfIndex	
		ipRouteMetric1	
		ipRouteMetric2	
		ipRouteMetric3	
		ipRouteMetric4	
		ipRouteNextHop	
		ipRouteType	
		ipRouteProto	
		ipRouteAge	
		ipRouteMask	
		ipRouteMetric5	
		ipNetToMediaIfIndex	
		ipNetToMediaPhysAddress	
		ipNetToMediaNetAddress	
		ipNetToMediaType	
		ipRoutingDiscards	
icmp	1.3.6.1.2.1.5	icmpInMsgs	Trata diversos aspectos dos pacotes ICMP, tais como descartados, com erros.
		icmpInErrors	
		icmpInDestUnreachs	
		icmpInRedirects	
		icmpInEchos	
		icmpInEchoReps	
		icmpOutMsgs	
		icmpOutErrors	
		icmpOutDestUnreachs	
		icmpOutRedirects	
		icmpOutEchos	
		icmpOutEchoReps	
tcp	1.3.6.1.2.1.6	tcpMaxConn	Trata diversos aspectos dos pacotes TCP, estados, conexões.
		tcpInSegs	
		tcpOutSegs	
		tcpRetransSegs	
		tcpInErrs	
udp	1.3.6.1.2.1.7	udpInDatagrams	Trata de estatísticas do UDP, datagramas recebidos, enviados, portos ocupados.
		udpNoPorts	
		udpInErrors	
		udpOutDatagrams	
		udpLocalAddress	
		udpLocalPort	

Tabela 2 - Informações retiradas da RFC1213-MIB

Nome da árvore iso.org. internet.mgmt. mib-2.(x)	**OID**	**Campos lidos**	**Descrição**
ifXEntry	.1.3.6.1.2.1.31.1.1.1	ifInMulticastPkts	Contém informação adicional das interfaces do equipamento gerido.
		ifInBroadcastPkts	
		ifOutMulticastPkts	
		ifOutBroadcastPkts	
		ifLinkUpDownTrapEnable	
		ifHighSpeed	
		ifPromiscuousMode	
		ifConnectorPresent	
		ifCounterDiscontinuityTime	

Tabela 3 - Informações retiradas da IF-MIB

3.4 Verifying and executing scripts

São os scripts **main_servers.sh**, cujos parâmetros de entrada são o tipo de serviço e o identificador do servidor (pois podem existir vários servidores com o mesmo serviço) e **main_equipment.sh**, com parâmetro de entrada o nome do equipamento. Estes são responsáveis por:

- No caso dos servidores, o script **main_servers.sh** verifica através do seu primeiro parâmetro de entrada qual o tipo de serviço que será processado, e manda executar o data processing script correspondente a este serviço, passando ao script correspondente o segundo parâmetro de entrada que recebeu, que é o identificador do servidor. De seguida, manda executar o sql script correspondente a este serviço.
- No caso dos equipamentos, o script **main_equipment.sh** através do seu parâmetro de entrada lê da base de dados, com o comando SELECT, de que equipamento se trata, se é um router ou um switch. Seguidamente manda executar o data processing script correspondente a este equipamento, passando ao script o mesmo parâmetro de entrada que recebeu, o nome do equipamento. A seguir manda executar um outro shell script e este é que por sua vez irá chamar o script .sql correspondente a este equipamento. Estes passos serão detalhados no sub-capítulo seguinte, mas deve-se ao facto de que um script.sql não aceita parâmetros de entrada.

3.4.1 Data Processing Scripts

Os data processing scripts são os responsáveis dentro desta arquitectura por processar e retirar os dados relevantes dos ficheiros de log e dos equipamentos da rede, através de comandos shell (ver Anexo IV).

Foram criados scripts para diversos serviços, treze no total - **script_apache.sh** – serviço Apache, **script_ftp.sh** – serviço ProFTPd, **script_dhcp.sh** – serviço DHCPd, **script_ssh.sh** – serviço OpenSSH, **script_samba.sh** – serviço Samba, **script_mail.sh** – serviço Sendmail, **script_nfs.sh** – serviço NFS, **script_win_web.sh** – serviço IISWeb, **script_win_ftp.sh** – serviço IISftp, **script_win_dhcp.sh** – serviço DHCP, **script_win_mail.sh** – serviço IISSMTP, **script_terminal.sh** – serviço Terminal, **script_file.sh** – serviço File Sharing - dentre eles alguns em máquinas Linux e outros em Windows. Recebem como parâmetro de entrada o identificador do servidor pois, numa rede pode haver mais do que uma máquina com o mesmo serviço. No capítulo 4 descrevem-se todos os serviços instalados, seus ficheiros de log, processing scripts e os dados obtidos destes ficheiros através destes.

Também foram criados scripts que limpam os dados lidos das MIBs dos equipamentos, o **script_router.sh** e o **script_switch.sh** para routers e switches respectivamente. Recebem como parâmetro de entrada o nome do equipamento (é suposto o administrador da rede ter configurado o nome do equipamento). Estes serão descritos no capítulo 5.

Em todos os processing data scripts é necessário ler e actualizar informações presentes na base de dados relativas aos serviços, filtrados pelo **id**, e relativas aos equipamentos, filtrados pelo nome do equipamento:

- Nos relativos aos serviços (**script_apache.sh**, **script_ftp.sh**, **script_dhcp.sh**, **script_ssh.sh**, **script_samba.sh**, **script_mail.sh**, **script_nfs.sh**, **script_win_web.sh**, **script_win_ftp.sh**, **script_win_dhcp.sh**, **script_win_mail.sh**, **script_terminal.sh**, **script_file)** é necessário ler o caminho em que está o ficheiro de log na máquina do administrador, e a data e hora que o script foi processado a última vez através do comando SELECT. Também são actualizadas, através do comando UPDATE, a data e hora que o script foi processado (que nada mais são do que a data e hora actual).
- Nos relativos aos equipamentos (**script_router.sh** e **script_switch.sh)** é necessário ler o **id** (identificador) do equipamento, e a data e hora que o script foi processado a última vez, através do comando SELECT. Também nestes são actualizadas, através do comando UPDATE, a data e hora que o script foi processado.

3.4.2 SQL Scripts

No caso dos servidores são scripts .sql (**script_base_dados_apache.sql, script_base_dados_ftp.sql, script_base_dados_dhcp.sql, script_base_dados_mail.sql, scrip_base_dados_samba.sql, script_base_dados_ssh.sql, script_base_dados_nfs.sql, script_base_dados_win_web.sql, script_base_dados_win_ftp.sql,**

script_base_dados_win_dhcp.sql, script_base_dados_win_mail.sql, script_base_dados_win_terminal.sql, script_base_dados_win_file.sql) responsáveis por:

- Criar, se não existir, a base de dados utilizando o comando CREATE DATABASE.
- Através do comando USE, indica ao MySQL qual base de dados será utilizada como padrão para as consultas subsequentes.
- Criar dentro da base de dados indicada, se não existir, uma tabela através do comando CREATE TABLE.
- Carregar os dados provenientes de um ficheiro de texto (o que foi criado anteriormente pelos processing data scripts) para a tabela através do comando LOAD DATA INFILE.

No caso dos equipamentos este processo é feito por shell scripts, **load_dados_router.sh** e **load_dados_switch.sh**, pois, scripts .sql não aceitam a passagem de parâmetros. Como é necessário passar o nome do equipamento como parâmetro para lhe ser criada uma base de dados e as suas tabelas, os passos seguidos foram:

- Criar, se não existir, a base de dados utilizando o comando CREATE DATABASE, mas da seguinte forma:

```
mysql -u $user -p$pass_mysql -e "CREATE DATABASE IF NOT EXISTS $1"
```

Figura 9 - Comando que cria a base de dados

Em que $1 é o nome do equipamento, logo, será criada uma base de dados com o seu nome. O -e significa **para executar**. É usado pois temos de utilizar o comando SQL embutido no comando shell.

- Executa-se o script .sql (**script_base_dados_router.sql** ou **script_base_dados_switch.sql**) da seguinte forma:

```
mysql -u $user -p$pass_mysql < /trabalho/script_base_dados_router.sql $1
```

Figura 10 - Comando shell que utiliza a base de dados especificada

Note que neste último comando o $1 não é um parâmetro de entrada. Apenas indica que será utilizada a base de dados $1. Neste script são criadas as tabelas dentro da base de dados $1, isto é, dentro da base de dados com o nome do equipamento.

- Carregam-se os dados para as tabelas da seguinte forma:

```
mysql -u $user -p$pass_mysql $1 -e "LOAD DATA INFILE
'$path/saida_script_"$1"/saida_"$1"_system.txt' INTO TABLE description"
```

Figura 11 - Comando para carregar os dados para tabelas indicadas

Neste caso é utilizado no nome dos ficheiros (e directórios) o parâmetro de entrada (nome do equipamento) pois é necessário saber quais ficheiros (de quais equipamentos) são necessários carregar.

A sintaxe dos comandos SQL é descrita no capítulo Base de dados e no Anexo VI.

3.5 Interface Web

Foi desenvolvida uma interface Web para uso do administrador da rede. O administrador, por meio desta, apenas deverá inserir os servidores e equipamentos de rede que quer que sejam monitorizados na rede. Haverá então todo o processamento dos outros blocos do sistema, anteriormente explicados neste capítulo, e após todo o processo é possível ao administrador ter uma visão unificada dos recursos e informações da rede que administra.

Esta interface será melhor explicada no capítulo 7 deste trabalho.

3.6 Síntese

Neste capítulo foi proposto um sistema integrado para recolha de informação de rede. Explicou-se em que consiste cada bloco do sistema desenvolvido neste trabalho e representado na figura 1. As principais noções adquiridas são a recolha remota dos dados distribuídos por diversos equipamentos e serviços na rede, o processamento dos dados, o armazenamento destes e a sua visualização através de uma interface *Web*.

4 Monitorização de serviços

4.1 Introdução

Servidor é um computador que numa rede administra e controla o acesso de outros computadores à totalidade ou às diferentes partes da rede. Este optimiza os respectivos recursos centralizando a informação, permitindo acréscimo de segurança e diminuição de custos. Surge assim a arquitectura mais utilizada nas redes de telecomunicações, a cliente-servidor. O servidor fornece os serviços e o cliente consome os serviços, havendo uma separação de funções.

Este capítulo apresenta os diversos serviços estudados neste trabalho e a análise dos respectivos ficheiros de log.

Durante a realização do trabalho, apenas nas ocorrências descritas neste documento, foi utilizado o software Webmin [Webmin2008]. Este apresenta uma interface *Web* para gerenciamento de ambientes Unix e Linux. Contém diversas funcionalidades tais como: administração de impressoras, sistema de boot, discos, partições, configuração de diversos serviços (Apache, Ftp, MySQL), gestão de quotas, utilizadores, grupos, entre outras.

4.2 Servidores instalados em máquina com sistema operativo Linux

Os serviços Linux estudados neste trabalho foram instalados numa máquina com sistema operativo Ubuntu 7.04 (Feisty Fawn) AMD64.

A International Data Corporation (IDC) anunciou directamente do Linux Foundation Collaboration Summit, um novo relatório sobre o mercado de servidores Linux. O estudo, intitulado "The Role of Linux Servers in Commercial Workloads", foi financiado pela Linux Foundation e descreve a tendência de mercado de servidores Linux, além de fazer uma previsão sobre o tema.

O movimento desse mercado em 2007 foi de US$ 21 bilhões. Espera-se para 2011 um volume de US$ 49 bilhões – mais que o dobro. O Linux está presente em toda a infra-estrutura de rede mundial, desde servidores *Web* até sistemas de arquivos, Domain Name System (DNS) e Dynamic Host Configuration Protocol (DHCP). Isso viabiliza o uso do sistema como a base de negócios do mercado, através de banco de dados, do Planeamento de Recursos da Empresa (Enterprise Resource Planning – ERP) e da Gerência de Relação com o Cliente (Customer relation management – CRM). De acordo com a IDC, uma parte significante desse crescimento está na migração de servidores Unix para servidores Linux [IDC2008].

Segue-se nos próximos sub-capítulos a descrição dos ficheiros de log de cada serviço Linux instalado e todo processamento feito sobre estes. No Anexo II há a descrição de como foi feita a instalação e configuração destes serviços.

4.2.1 Apache

É o servidor HTTP (*Web*) número um da Internet. É um esforço para desenvolver e manter um servidor HTTP de código aberto para modernos sistemas operativos incluindo UNIX e Windows. Proporciona segurança, eficiência, extensibilidade. Apache é o mais popular servidor na Internet desde Abril de 1996 e é um projecto da Apache Software Foundation. Escuta no porto 80 e a versão deste serviço utilizada neste trabalho foi a 2.2.4.

4.2.1.1 Ficheiros de logs

O ficheiro de log **access_log** deste servidor localiza-se em **/usr/local/apache2/logs** e armazena todos os pedidos feitos ao servidor. Os dois formatos típicos para o ficheiro de log **access_log** são:

Nome	**Formato**
common	"%h %l %u %t \"%r\" %>s %b"
combined	"%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\""

Tabela 4 - Formatos típicos log Apache

O significado de cada parâmetro, tomando como exemplo a linha seguinte retirada da página do Apache na sessão correspondente ao formato dos logs [ApacheLogs2008], é descrito na tabela seguinte:

127.0.0.1 – frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif" HTTP/1.0" 200 2326 "http://www.example.com/start.html**" "Mozilla/4.08 [en] (Win98;I;Nav)"**

Parâmetro	**Significado**	**Exemplo**
%h	ip do cliente. Se existir um servidor proxy entre o cliente e o servidor, este endereço será o do proxy.	127.0.0.1
%l	Identidade RFC1213 do cliente	-
%u	quando é pedida autenticação http pelo servidor apache, esta directiva fornece o userid da pessoa que fez pedido. Quando o documento não é protegido por palavra pass este campo apresenta um hífen.	frank
%t	o tempo que o servidor acabou de processar o pedido feito. O número seguinte corresponde à *time zone* do cliente	[10/Oct/2000:13:55:36 -0700]

\"%r\"	o pedido feito pelo cliente	"GET /apache_pb.gif" HTTP/1.0"
%>s	o status do servidor enviado ao cliente. Lista detalhada dos códigos de status encontra-se na RFC 2616 secção10	200
%b	o tamanho do objecto retornado ao cliente	2326
\"%{Referer}i\"	é o site que o cliente referenciou no pedido	"http://www.example.com/start.html"
"%{User-agent}i\"	informações do browser do cliente	"Mozilla/4.08 [en] (Win98;I;Nav)"

Tabela 5 - Parâmetros Apache e seus significados

Os hífenes surgem nos logs quando a saída de um determinado parâmetro não está disponível.

Por defeito o formato que vem configurado é o *common*. No âmbito deste trabalho o formato uilizado foi o *combined* pois, este apresenta um maior número de informações, como visto na tabela 4. Para esta mudança substitui-se o comando **CustomLog logs/access_log common** pelo comando **CustomLog logs/access_log combined** no ficheiro de configuração.

Na figura seguinte observa-se um trecho de um ficheiro de log do serviço Apache, o **access_log**:

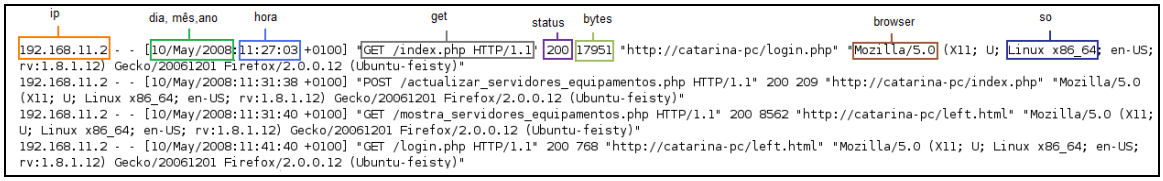


Figura 12 - access_log

4.2.1.2 Apache Data Processing Script

Através do **script_apache.sh** fez-se o processamento dos dados registados nos ficheiros de log que são importantes e que queremos armazenar. No ficheiro de texto de saída, **saida_apache.txt**, os dados devem estar em colunas e separados por tabulações de forma a que a base de dados possa interpretar e carregar correctamente os dados através do comando LOAD DATA INFILE (sintaxe no Anexo VI). A seguir apresenta-se a descrição de cada coluna do ficheiro **saída_apache.txt**, e o significado dos dados que o constituí:

Coluna correspondente no ficheiro de saída	**Notação utilizada no script para os dados retirados do log**	**Significado**
1	vazia	Receberá o identificador da linha na base de dados
2	id_server	é o identificador do serviço numa

		determinada máquina. Recebe o valor do parâmetro de entrada, $1.
3	ano-mes-dia	data que cliente acedeu ao serviço. Fica no formato DATE (yyyy-mm-dd) do MySQL.
4	hora	hora que cliente acedeu ao serviço
5	ip	IP dos clientes
6	get	pedido feito pelo cliente
7	status	status do servidor
8	browser	browser utilizado pelos cliente
9	so	sistema operativo na máquina do cliente
10	bytes	número de bytes dos objectos retornados aos clientes

Tabela 6 - Dados processados Apache

Através da figura 12 observam-se as posições de cada elemento descrito na tabela acima. A seguir descrevem-se alguns dos comandos utilizados neste processamento:

```
ano=(`cat $path |cut -d"/" -f 3 | cut -d":" -f 1`)

mes=(`cat $path |cut -d"/" -f 2 `)

dia=(`cat $path | cut -d"[" -f 2 | cut -d"/" -f 1`)

hora=(`cat $path | cut -d"/" -f 3 |cut -c6-13`)

ip=(`cat $path | cut -d" " -f 1`)

get=cat $path | cut -d"\"" -f 2  > $path_auxiliar1

status=(`cat $path | cut -d" " -f 9`)

browser=(`cat $path | cut -d"\"" -f 6 | cut -d" " -f 1`)

so= cat $path | cut -d";" -f 3 > $path_auxiliar2

bytes=(`cat $path | cut -d" " -f 10`)
```

NOTAS:

O **mês** vem com a notação Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec. Com um **case** (ver formato no Anexo I) transforma-se para o número correspondente a cada mês (de 01 a 12).

Para os casos **get** e **so** a informação foi encaminhada para ficheiros auxiliares pois estes apresentam mais do que um campo de resultado. Por exemplo, o campo **so** descrito na figura

seguinte - 1ªlinha, é Linux x86_64, logo apresenta dois campos e um espaço pelo meio. Como o processamento é feito através de arrays, em que os campos são separados por espaços, este exemplo é considerado como se tivessem dois elementos, ao invés de um. Depois de estarem em ficheiros auxiliares, os campos são cortados através do comando **so[i]=`cat $path_auxiliar2 | tail -n $y | head -n 1 `,** em que **y** é o tamanho do ficheiro path_auxiliar2 (extraído com o comando **(`cat $path_auxiliar2 | wc -l`**)) e decresce de uma unidade a cada interacção.

Figura 13 - Notas Apache

Na figura a seguir podem-se observar os dados no ficheiro de saída:

```
        1       2008-05-10      11:27:03        192.168.11.2    GET /index.php HTTP/1.1 200     Mozilla/5.0     Linux x86_64    17951
        1       2008-05-10      11:31:38        192.168.11.2    POST /actualizar_servidores_equipamentos.php HTTP/1.1   200     Mozilla/5.0     Linux
x86_64  209
        1       2008-05-10      11:31:40        192.168.11.2    GET /mostra_servidores_equipamentos.php HTTP/1.1        200     Mozilla/5.0     Linux
x86_64  8562
        1       2008-05-10      11:41:40        192.168.11.2    GET /login.php HTTP/1.1 200     Mozilla/5.0     Linux x86_64    768
```

Figura 14 - saída_apache.txt

4.2.2 ProFTPD

É um serviço de acesso de utilizadores a um disco ou directório através do protocolo de transporte FTP (File Transfer Protocol). Disponível para sistemas operativos Linux e Unix. Surgiu com o intuito de ser um servidor FTP seguro e confiável. Escuta no porto 21 e a versão utilizada foi a 1.3.

4.2.2.1 Ficheiros de logs

Apresenta dois ficheiros de log, o TransferLog que fica em **/var/log/proftpd/xferlog** e o SystemLog em **/var/log/proftpd/proftpd.log**.

O **xferlog** contém os campos separados por espaços e na ordem descrita na tabela:

Campos	**Significado**
current-time	"DDD MMM dd hh:mm:ss YYYY"- descreve o dia da semana, mês, dia do mês, hora, minuto, segundo e o ano
tranfer-time	tempo em segundos da transferência
remote-host	o nome do computador remoto
file-size	tamanho em bytes do ficheiro transferido
filename	o nome do ficheiro transferido (caminho completo)
tranfer-type	indica o tipo de transferência: letra **a** representa ascii e letra **b** representa binária
special-action-flag	indica através do carácter se : **C** ficheiro comprimido, **U** não comprimido, _ sem identificação
direction	é a direcção da transferência: **o** outgoing, **i** incoming, **d** deleted
access-mode	indica o método do login: **a** se usuário for anonymous, **g** se for pedida password, **r** se for um usuário local.

username	é o nome do usuário
service-name	nome do serviço, neste caso FTP
authentication-method	método de autenticação: **0** é quando não há autenticação e **1** é quando é feita pela RFC931
authentication-user-id	é o id do usuário. Usa-se * quando não esta disponível
completion-status	é o estado da transferência: c se foi completa e i se incompleta

Tabela 7 - Parâmetros xferlog e seus significados

Na figura abaixo encontra-se um trecho deste ficheiro de log:

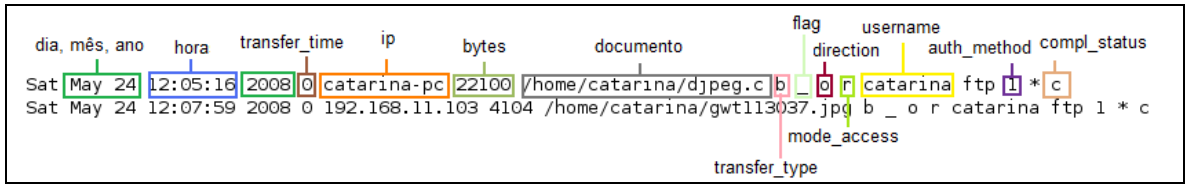


Figura 15 - xferlog

O outro log, **proftpd.log**, segue o formato SystemLog. Estes logs são criados pelo serviço de log de dados **Syslog**, descrito na RFC3164 [RFC3164 2008].

O formato das mensagens registadas apresenta três partes: **PRI**, **HEADER** e **MSG**, porém os registos encontrados não apresentam o PRI. O **HEADER** contém dois campos:

- TIMESTAMP: é o dia e hora local. Apresenta o formato "Mmm dd hh:mm:ss". Mmm é o mês - Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec - ; dd é o dia do mês (se for menor que 10, há um espaço antes do número, logo, há dois espaços entre o mês e o dia); hh:mm:ss é a hora local e é representada com formato de 24 horas.
- HOSTNAME: contém o h*ostname*, ou seu endereço IPv4 ou IPv6.

O **MSG** contém dois campos:

- TAG: contém o nome do serviço ou programa que gera as mensagens, o identificador (entre parêntesis rectos).
- CONTENT: detalhes das mensagens. Informação depois dos ":".

Na figura abaixo encontra-se um trecho deste ficheiro de log:

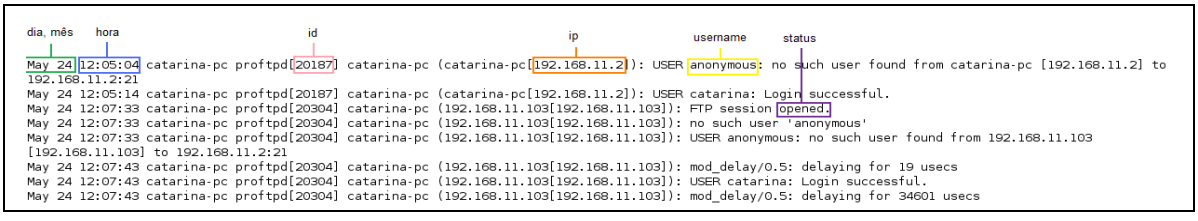


Figura 16 - proftpd.log

4.2.2.2 ProFTPD Data Processing Script

Através do **script_ftp.sh** fez-se o processamento dos dados registados nos ficheiros de log **proftpd.log** e **xferlog** que são importantes e que se pretendem armazenar. Para um melhor processamento, através do auxílio do comando **grep**, dividiram-se em dois casos os dados do primeiro ficheiro: linhas que apresentam a palavra **session** e linhas que apresentam a palavra **USER**. No ficheiro de texto de saída, **saida_ftp.txt**, os dados devem estar em colunas e separados por tabulações para a base de dados poder interpretar e carregar os dados através do comando LOAD DATA INFILE duma maneira correcta.

Na tabela seguinte apresenta-se a descrição do **script_ftp.sh**:

Coluna correspondente no ficheiro de saída	**Notação utilizada no script para os dados retirados do log**	**Qual ficheiro de log contém esta informação**	**Significado**
1	vazia	-	Receberá o identificador da linha na base de dados
2	id_server	-	é o identificador do serviço numa determinada máquina. Recebe o valor do parâmetro de entrada, $1
3	ano-mes-dia	proftpd.log (não apresenta o ano) / xferlog	data que cliente acedeu ao serviço. Fica no formato DATE (yyyy-mm-dd) do MySQL
4	hora	proftpd.log / xferlog	hora que cliente acedeu ao serviço
5	id	proftpd.log	identifica a sessão
6	ip	proftpd.log / xferlog	IP do cliente ou nome da máquina cliente
7	username	proftpd.log / xferlog	identifica o utilizador
8	status	proftpd.log	status logon. Pode ser opened, closed, nothing
9	documento	xferlog	caminho completo do documento da transferência
10	bytes	xferlog	número de bytes dos ficheiros transmitidos
11	transfer_time	xferlog	tempo em segundos da transferência
12	transfer_type	xferlog	tipo da transferência
13	flag	xferlog	como esta armazenado
14	direction	xferlog	direcção da transferência
15	mode_access	xferlog	método do login

16	auth_method	xferlog	método de autenticação
17	compl_status	xferlog	estado da transferência

Tabela 8 - Dados processados Proftpd

Através das figuras 15 e 16 observam-se as posições de cada elemento descrito na tabela anterior. De seguida descrevem-se alguns dos comandos utilizados neste processamento:

proftpd.log:

mes=(`cat $path | grep *palavra* | cut -d" " -f 1 `)

dia=(`cat $path | grep *palavra* | cut -d" " -f 2 `)

hora=(`cat $path | grep *palavra* | cut -d" " -f 3 `)

id=(`cat $path | grep *palavra* | awk '{print $5}' | cut -d"[" -f 2 | cut -d"]" -f 1 `)

ip=(`cat $path | grep *palavra* | awk '{print $7}' | cut -d"(" -f 2 | cut -d"[" -f 2 | cut -d"]" -f 1`)

username=(`cat $path | grep USER | awk '{print $9}'| cut -d":" -f 1 `)

status=(`cat $path | grep session | awk '{print $10}' | cut -d"." -f 1`)

status=(`cat $path | grep USER | awk '{print $10}' `)

xferlog: O comando para extração dos dados é sempre o mesmo e apresentado de seguida:

dados=(`cat $path | awk '{print $**x**}' `)

Figura 17 - Comandos extrair dados xferlog

O que muda é apenas a posição, descrita na tabela seguinte:

dados	**posição(x)**
ano	5
mes	2
dia	3
hora	4
ip	7
username	14
documento	9
byte	8
transfer_time	6
tranfer_type	10
flag	11
direction	12
mode_access	13
auth_method	16
compl_status	18

Tabela 9 - Posição elementos xferlog

NOTAS:

palavra leia-se session ou USER.

O **mês** nos dois ficheiros vem com a notação Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec. Com um **case** (ver formato no Anexo I) transforma-se para o número correspondente a cada mês (de 01 a 12).

Para os casos **tranfer_type, flag, direction, mode_access, auth_method, compl_status** utiliza-se um **case** para atribuir ao símbolo o seu significado correspondente descrito na tabela 7.

Figura 18 - Notas Proftpd

Como se efectuou o processamento em várias partes, no fim ordenaram-se os dados de forma que haja uma melhor visualização na base de dados. O comando a seguir ordena analisando coluna a coluna:

```
cat $path_saida_desordenado | sort –k 1 > $path_saida
```

Figura 19 - Ordenação por data e hora

Após o tratamento total dos dados estes ficam conforme a figura seguinte:

```
        2       2008-05-24      12:05:04        20187   192.168.11.2    anonymous       nothing
        2       2008-05-24      12:05:04        20187   192.168.11.2            opened
        2       2008-05-24      12:05:14        20187   192.168.11.2    catarina        Login
        2       2008-05-24      12:05:16                catarina-pc     catarina                /home/catarina/djpeg.c  22100   0       binary
without_identification  outgoing        user_local      RFC931  tranfer_complete
        2       2008-05-24      12:07:33        20304   192.168.11.103  anonymous       nothing
        2       2008-05-24      12:07:33        20304   192.168.11.103          opened
        2       2008-05-24      12:07:43        20304   192.168.11.103  catarina        Login
        2       2008-05-24      12:07:59                192.168.11.103  catarina                /home/catarina/gwt113037.jpg    4104    0       binary
without_identification  outgoing        user_local      RFC931  tranfer_complete
```

Figura 20 - saída_ftp.txt

4.2.3 DHCPd

Dynamic Host Configuration Protocol (DHCP), definido pela RFC2131, é um serviço de rede que permite que as máquinas recebam as configurações de rede a partir de um servidor - conceito de aluguer de endereços. Quando o cliente quer receber configurações faz *broadcast* no porto UDP 68 e o servidor responde no porto UDP 67.

As configurações mais comuns fornecidas por um servidor DHCP a clientes DHCP incluem: endereço ip e máscara de rede, DNS, WINS. Contudo, o servidor também pode disponibilizar propriedades como: nome do anfitrião e de domínio, porta de ligação por omissão, *default gateway*, servidor horário.

A vantagem de utilizar um servidor DHCP na rede é a de automatizar as alterações na rede. As alterações apenas são efectuadas nas configurações no servidor e todas as máquinas da rede são reconfiguradas automaticamente. Torna-se mais simples adicionar computadores na rede já que não

é necessário verificar a disponibilidade de um endereço IP, não havendo assim conflitos nessa atribuição.

O servidor pode fornecer configurações por dois métodos:

- Endereço MAC: usa o DHCP para identificar endereços únicos de hardware e indicar-lhes uma configuração única.
- Gama de endereços Address Pool: implica definir uma gama de endereços IP a partir da qual os clientes DHCP são fornecidos, com propriedades e configurações dinâmicas. Quando o cliente DHCP abandona a rede, a configuração expira e o endereço é devolvido à gama de endereços disponíveis, e poderá assim ser utilizado por outro cliente DHCP.

O servidor e versão utilizada neste trabalho são respectivamente ISC DHCPd 3.0.4 e o porto é o 67.

4.2.3.1 Ficheiros de logs

A informação referente aos ficheiros de log deste serviço não fica registada em nenhum ficheiro específico, mas sim junto com outras informações do sistema (kernel, cron, entre outros) em **/var/log/messages** ou em **var/log/syslog**. Neste último também é possível observar informações de log do cliente, mas como no âmbito deste trabalho apenas nos interessa analisar as informações do servidor, utilizaremos o ficheiro que está presente em **/var/log/messages**.

Estes ficheiros de log também são criados pelo serviço de log de dados **Syslog**, descrito na RFC3164 [RFC3164 2008] e seguem o formato referenciado anteriormente no sub-capítulo 4.2.2.1.

Na figura abaixo encontra-se um trecho deste ficheiro de log:

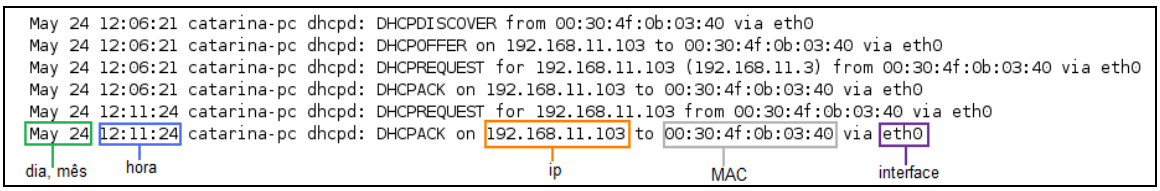


Figura 21 - messages

4.2.3.2 DHCPd Data Processing Script

Apesar do protocolo de aluguer apresentar quatro fases: *discover, offer, request* e *acknowledge*, apenas se retirou informação do ficheiro de log correspondente ao *acknowledge*, que corresponde a fase final e "definitiva" do processo.

Através do **script_dhcp.sh** fez-se o processamento dos dados registados nos ficheiros de log. No ficheiro de texto de saída, **saida_dhcp.txt**, os dados devem estar em colunas e separados por tabulações de forma a que a base de dados possa interpretar e carregar os dados através do

comando LOAD DATA INFILE correctamente. Na tabela seguinte apresenta-se a descrição do **script_dhcp.sh**:

Coluna correspondente no ficheiro de saída	**Notação utilizada no script para os dados retirados do log**	**Significado**
1	vazia	Receberá o identificador da linha na base de dados
2	id_server	é o identificador do serviço numa determinada máquina. Recebe o valor do parâmetro de entrada, $1.
3	mes-dia. Não apresenta informação do ano(declara-se).	data que cliente acedeu ao serviço . Armazena-se no formato DATE yyyy-mm-dd do MySQL
4	hora	hora que houve atribuição do endereço
5	ip	IP dos cliente
6	mac	endereço MAC da interface do cliente
7	interface	interface que o servidor propaga

Tabela 10 - Dados processados DHCPd

Os dados retirados, como dito anteriormente, correspondem às linhas do processo *acknowledge*. Estas linhas são as que apresentam a palavra **DHCPACK** e a palavra **on** (porque haviam outras linhas cuja informação era repetida).

Através da figura 21 observam-se as posições dos dados retirados e a seguir descrevem-se alguns dos comandos utilizados neste processamento.

Os campos **dia**, **mes**, **hora**, **ip**, **mac**, **interface** foram filtrados com o mesmo comando, mudando apenas a posição destes. O comando e suas posições apresentam-se de seguida:

```
dados=(`cat $path | grep DHCPACK |  grep on | awk '{print $x}' `)
```

Figura 22 - Comando extrair dados DHCPd

dados	**posição(x)**
mes	1
dia	2
hora	3
ip	8
mac	10
interface	12 e 13

Tabela 11 - Posição dos dados no log do DHCPd

Além do comando da figura 22, do campo **mac** são retirados os caracteres **":"** através do comando **sed** (ver tabela seguinte); e do campo interface como foram extraídos os campos da posição 12 e 13 (uma vez que o valor da interface por vezes aparecia na posição 12 e outras vezes na 13), corta-se com o **cut** conforme tabela seguinte, pois no pior caso o resultado seria **via eth0**.

mac	(`cat $path | grep DHCPACK | grep on | awk '{print $10}'| sed '/\:/s///g'`)
interface	(`cat $path | grep DHCPACK | grep on | awk '{print $12 $13}' | cut -d"a" -f 2 `)

Tabela 12 - Pipes adicionais DHCPd

NOTA:

O **mês** vem com a notação Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec. Com um **case** (ver formato no Anexo I) transforma-se para o número correspondente a cada mês (de 01 a 12).

Figura 23 - Notas DHCPd

Na figura seguinte apresenta-se o ficheiro de log depois de limpo:

```
3       2008-05-24      12:06:21        192.168.11.103  00304f0b0340    eth0
3       2008-05-24      12:11:24        192.168.11.103  00304f0b0340    eth0
```

Figura 24 - saída_dhcp.txt

4.2.4 OpenSSH

Secure Shell, SSH, é um serviço que permite administrar máquinas remotamente. Permite executar aplicativos gráficos, transferir arquivos, encapsular outros protocolos, através de um túnel seguro. Uma das grandes vantagens deste serviço, em relação a outros como por exemplo o Telnet, é que apresenta grande ênfase na segurança pois utiliza um conjunto de técnicas de criptografia assegurando que apenas as pessoas autorizadas terão acesso ao servidor, que todos os dados transmitidos sejam impossíveis de decifrar e que a integridade da conexão seja mantida. Isto é possível pois a comunicação entre cliente e servidor é feita de forma encriptada usando chaves públicas/privadas RSA.

Para testar o funcionamento do servidor utilizaram-se os seguintes recursos:

- Num computador cliente com sistema operativo Windows utilizou-se o programa PUTTY, que é um cliente SSH para Windows (também pode funcionar como cliente telnet). Introduz-se apenas o ip do servidor e o porto que este escuta. Será também pedido ao utilizador o username e password.
- Num computador cliente com sistema Linux utilizou-se um dos seguintes comandos:

ssh -l username ip_servidor	**ssh username@ip_servidor**

Figura 25 - Comandos conexão SSH

Como se observa nos comandos anteriores, o username (nome do utilizador do servidor) deve ser passado, caso contrário, o SSH tentaria conectar-se à estação remota utilizando a mesma conta do computador local. Na primeira conexão também é questionado se deseja adiciona-la à lista de *hosts* conhecidos.

A versão utilizada neste trabalho foi a OpenSSH_4.3, mas existem outras como o Tecnia e o SunSSH. A utilizada possibilita além de ssh: rcp com scp (utilizado neste trabalho, ver capítulo Arquitectura do sistema) e ftp com sftp. Escuta no porto 22.

4.2.4.1 Ficheiros de logs

Os ficheiros de log deste servidor também apresentam o formato SystemLog, referenciado anteriormente no sub-capítulo 4.2.2.1.

Na figura a seguir pode-se observar um trecho do ficheiro de log, **auth.log**:

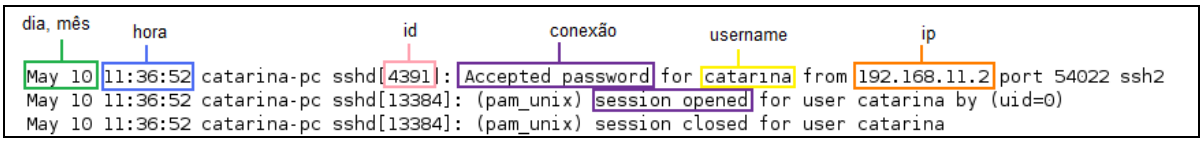


Figura 26 - auth.log

4.2.4.2 OpenSSH Data Processing Script

Através do **script_ssh.sh** fez-se o processamento dos dados registados nos ficheiros de log. Como o conteúdo das linhas é diferente, fez-se o processamento em quatro partes e no fim ordenaram-se de forma a possibilitar uma melhor visualização na base de dados. No ficheiro de texto de saída, **saida_ssh.txt**, os dados devem estar em colunas e separados por tabulações, pelo mesmo motivo referido anteriormente, de maneira a que a base de dados possa interpretar e carregar os dados através do comando LOAD DATA INFILE. Na tabela seguinte apresenta-se a descrição do **script_ssh.sh**:

Coluna correspondente no ficheiro de saída	**Notação utilizada no script para os dados retirados do log**	**Significado**
1	vazia	Receberá o identificador da linha na base de dados
2	id_server	é o identificador do serviço numa determinada máquina. Recebe o valor do parâmetro de entrada, $1
3	mes-dia. Não apresenta o ano	data que cliente acedeu ao serviço. Armazena-se no formato DATE yyyy-mm-dd do MySQL
4	hora	hora que cliente acedeu ao serviço ssh
5	username	*username* do utilizador
6	id	identificador da sessão
7	ip	IP do cliente
8	conexao	status da conexão. Pode ser: sessionopened; sessionclosed; Acceptedpassword

Tabela 13 - Dados processados OpenSSH

Os dados foram separados em quatro partes, para quatro ficheiros, consoante a informação das linhas. Estas partes foram repartidas consoante a tabela seguinte:

cat $path | grep sshd | egrep -v listening | grep password | egrep -v invalid > $path1
cat $path | grep sshd | egrep -v listening | grep session > $path2
cat $path | grep sshd | egrep -v listening | grep Failed | grep invalid > $path3
cat $path | grep sshd | egrep -v listening | grep Invalid > $path4

Tabela 14 - Separação log OpenSSH

O comando utilizado para limpar os logs é o da figura seguinte (path varia entre {path1, path2, path3, path4}):

```
dados=(`cat $path | awk '{print $x}' `)
```

Figura 27 - Comando extrair dados OpenSSH

Porém, de ficheiro para ficheiro a posição destes varia, e nem todos apresentam os mesmos dados. Isto resume-se na tabela seguinte:

dados	$path1	$path2	$path3	$path4
mes	1	1	1	1
dia	2	2	2	2
hora	3	3	3	3
username	9	11	11	8
id	5	5	5	5
ip	11	Não apresenta esta informação	13	10
conexao	6 e 7	7 e 8	6 e 7	6 e 7

Tabela 15 - Posição dos dados no log do OpenSSH

O campo **id** em todos os ficheiros além de utilizar este comando ainda deve ser filtrado através do seguinte pipe de maneira a retirar-lhe os parêntesis rectos:

```
id=(`cat $path4 | awk '{print $5}' | cut -d"[" -f 2 | cut -d"]" -f 1 `)
```

Figura 28 - Pipe adicional para o campo id

NOTA:

O **mês** vem com a notação Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec. Com um **case** (ver formato no Anexo I) transforma-se para o número correspondente a cada mês (de 01 a 12).

Figura 29 - Notas OpenSSH

O ficheiro de saída **saida_ssh.txt** apresenta-se da seguinte forma:

```
6    2008-05-10    11:36:52    catarina    4391     192.168.11.2    Acceptedpassword
6    2008-05-10    11:36:52    catarina    13384            sessionclosed
6    2008-05-10    11:36:52    catarina    13384            sessionopened
```

Figura 30 - saída_ssh.txt

4.2.5 Samba

Permite que máquinas Linux compartilhem unidades de disco e impressoras com máquinas Windows. Criado por Andrew Tridgell, natural da Austrália, quando ainda era estudante. O Samba nasceu a partir de um problema: o de integrar um servidor Unix com um PC com DOS. Possuía uma versão NFS, que permitia o acesso aos directórios, mas o problema é que uma de suas aplicações exigia interface NetBios. A solução encontrada por Tridgell não foi simples. Ele escreveu um *sniffer* que permitisse analisar o tráfego de dados gerado pelo protocolo NetBios, fez engenharia reversa no protocolo SMB (Server Message Block) e o implementou no UNIX. Assim fez com que o servidor Unix aparecesse como um servidor de arquivos Windows em seu PC com DOS.

O seu código foi disponibilizado publicamente em 1992 por Tridgell, mas entretanto o projecto foi posto de lado até que a uns anos Tridgell resolveu voltar a dedicar-se ao projecto e descobriu que as documentações do protocolo SMB e Network Basic Input Output System (NetBios) estavam actualizadas. Porém uma empresa entrou em contacto com ele, reivindicando os direitos sobre o nome usado até então. Diante disso Tridgell encontrou o termo SAMBA (que tem as letras s, m, b). A partir daí o projecto cresceu.

É provido de grande segurança e permite que a sua configuração seja feita por meio de computadores remotos. É um software livre e está sob a licença GNU. A versão utilizada neste trabalho é a 3.024 e o porto que o serviço escuta é o 139.

4.2.5.1 Ficheiros de logs

Na secção [global] do ficheiro de configuração, na parte correspondente ao Debugging/Accounting é descrito a localização do ficheiro de log do servidor, que fica em **/var/log/samba/log.%m** (%m é o nome da máquina que acedeu ao serviço), o tamanho máximo do ficheiro de log (configurado para 10000Kb), e se quer que as informações também fiquem presentes no syslog (***yes/no***), opção escolhida neste trabalho é ***no***.

Como anteriormente descrito, a notação utilizada para armazenar os ficheiros de log do Samba é o nome da máquina cliente ou ip do cliente que acedeu ao serviço.

O nivel de logging, isto é, a quantidade de dados que pode ser registada nos ficheiros de log, pode ser configurado. Pode variar entre 1 e 10, e quanto maior o nível, maior o nível de detalhes. Se for configurado nível 0, nenhum registo é feito. Se não for configurado no ficheiro **/etc/samba/smb.conf**, o nível padrão é o 1. Normalmente não é preciso utilizar um nível maior do que 3.

Neste trabalho, optou-se por utilizar o nível padrão 1 pois, analisando a informação que é adicionada ao registo com níveis superiores (2 e 3), chegou-se à conclusão que não seria útil no âmbito deste trabalho. O nível três além de ocupar mais espaço em disco, torna o servidor mais lento, por isso só deve ser usado mesmo em casos específicos. Na tabela seguinte, pode-se comparar estes três níveis [Samba2007]:

Níveis	**Logs**
1	05/25/02 22:02:11 server (192.168.236.86) connect to service public as user pcguest (uid=503,gid=100) (pid 3377)
2	Got SIGHUP Processing section "[homes]" Processing section "[public]" Processing section "[temp]" Allowed connection from 192.168.236.86 (192.168.236.86) to IPC$ Allowed connection from 192.168.236.86 (192.168.236.86) to IPC/
3	05/25/02 22:15:09 Transaction 63 of length 67 switch message SMBtconX (pid 3377) Allowed connection from 192.168.236.86 (192.168.236.86) to IPC$ ACCEPTED: guest account and guest ok found free connection number 105 Connect path is /tmp chdir to /tmp chdir to / 05/25/02 22:15:09 server (192.168.236.86) connect to service IPC$ as user pcguest (uid=503,gid=100) (pid 3377) 05/25/02 22:15:09 tconX service=ipc$ user=pcguest cnum=105 05/25/02 22:15:09 Transaction 64 of length 99 switch message SMBtrans (pid 3377) chdir to /tmp trans <\PIPE\LANMAN> data=0 params=19 setup=0 Got API command 0 of form <WrLeh> <B13BWz> (tdscnt=0,tpscnt=19,mdrcnt=4096,mprcnt=8) Doing RNetShareEnum RNetShareEnum gave 4 entries of 4 (1 4096 126 4096) 05/25/02 22:15:11 Transaction 65 of length 99 switch message SMBtrans (pid 3377) chdir to / chdir to /tmp trans <\PIPE\LANMAN> data=0 params=19 setup=0 Got API command 0 of form <WrLeh> <B13BWz> (tdscnt=0,tpscnt=19,mdrcnt=4096,mprcnt=8) Doing RNetShareEnum RNetShareEnum gave 4 entries of 4 (1 4096 126 4096) 05/25/02 22:15:11 Transaction 66 of length 95 switch message SMBtrans2 (pid 3377) chdir to / chdir to /pcdisk/public call_trans2findfirst: dirtype = 0, maxentries = 6, close_after_first=0, close_if_end = 0 requires_resume_key = 0 level = 260, max_data_bytes = 2432

	unix_clean_name [./DESKTOP.INI] unix_clean_name [desktop.ini] unix_clean_name [./] creating new dirptr 1 for path ./, expect_close = 1 05/25/02 22:15:11 Transaction 67 of length 53 switch message SMBgetatr (pid 3377) chdir to /

Tabela 16 - Níveis de login samba

Na figura abaixo encontra-se um trecho deste ficheiro de log:

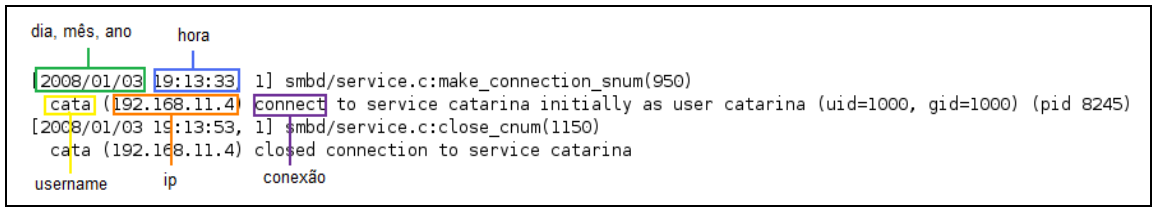


Figura 31 - log.cata

4.2.5.2 Samba Data Processing Script

Através do **script_samba.sh** fez-se o processamento dos dados registados nos ficheiros de log **log.%m**. No ficheiro de texto de saída, **saida_samba.txt**, os dados estão em colunas separados por tabulações. Na tabela seguinte apresenta-se a descrição do **script_samba.sh**:

Coluna correspondente no ficheiro de saída	**Notação utilizada no script para os dados retirados do log**	**Significado**
1	vazia	Receberá o identificador da linha na base de dados
2	id_server	é o identificador do serviço numa determinada máquina. Recebe o valor do parâmetro de entrada, $1
3	data	data que cliente acedeu ao serviço . Armazenar no formato DATE yyyy-mm-dd do MySQL
4	hora	hora que cliente acedeu ao serviço
5	ip	IP do cliente
6	nome_maquina	nome da máquina que acedeu ao samba
7	conexão	status da conexão. Pode ser connect ou closed

Tabela 17 - Dados processados Samba

Através da figura 31 observam-se as posições de cada elemento. Foram seleccionadas apenas as linhas com a palavra **service** (há outras informações sem importância, do sistema, sem esta palavra porém não foi demonstrado na figura 31) e a seguir descrevem-se os comandos utilizados neste processamento:

```
data=(`cat $path | grep service | cut -d" " -f 1 | cut -d"[" -f 2 | grep / | sed "/\/s//-/g"`)
```

```
hora=(`cat $path | grep service | grep / | cut -d" " -f 2 | cut -d"," -f 1 `)

ip=(`cat $path | grep service | cut -d"(" -f 2 | grep -e "\." | cut -d")" -f 1`)

nome_maquina[i]=`cat $path | grep service | grep ${ips[$i]} | cut -d"(" -f 1 | head -n 1 `

conexao=(`cat $path | grep service | cut -d"(" -f 2 | grep -e "\." | cut -d" " -f 2`)
```

Na figura seguinte observam-se os dados do ficheiro **saida_samba.txt**:

```
4       2008-01-03      19:13:33        192.168.11.4    cata    connect
4       2008-01-03      19:13:53        192.168.11.4    cata    closed
```

Figura 32 - saida_samba.txt

4.2.6 NFS

Network File System (NFS), desenvolvido pela Sun Microsystems, é agora suportado na maioria dos computadores. Baseado em RPC, UDP, e *stateless servers*. Foi desenvolvido com a finalidade de se poder montar partições ou directórios remotos como se pertencessem ao disco local e permite configurar diferentes permissões de acesso para cada cliente. É útil nos casos de ser necessário compartilhar poucos directórios numa rede pequena com apenas computadores GNU/Linux, podendo ser de configuração mais fácil quando comparado ao Samba. A versão instalada é a V4 e o porto ocupado é o 2049.

4.2.6.1 Ficheiros de logs

Os ficheiros de log desta versão localizam-se em **/var/log/daemon.log**, o que é um inconveniente pois, são registados juntamente com os logs de outros daemons, como por exemplo o MySQLd, DHCPd.

Este log também é criado pelo serviço de log de dados **Syslog**, descrito na RFC3164 [RFC3164 2008] e segue o formato referenciado anteriormente no sub-capítulo 4.2.2.1. A seguir apresenta-se um trecho do **daemon.log**:

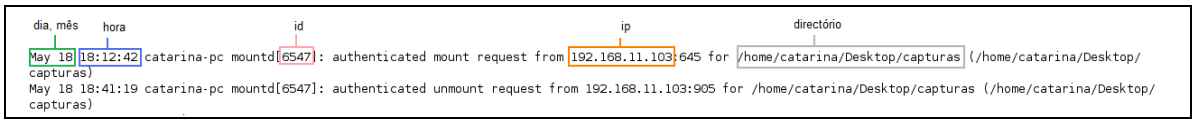


Figura 33 - daemon.log

4.2.6.2 Nfs Data Processing Script

Através do **script_nfs.sh** fez-se o processamento dos dados registados nos ficheiros de log. No ficheiro de texto de saída, **saida_nfs.txt**, os dados devem estar em colunas e separados por tabulações para que a base de dados possa interpretar e carregar os dados através do comando LOAD DATA INFILE de forma correcta. Na tabela seguinte apresenta-se a descrição do **script_nfs.sh**:

Coluna correspondente no ficheiro de saída	Notação utilizada no script para os dados retirados do log	Significado
1	vazia	Receberá o identificador da linha na base de dados
2	id_server	é o identificador do serviço numa determinada máquina. Recebe o valor do parâmetro de entrada, $1
3	mes, dia. Não apresenta informação do ano (define-se)	data que cliente acedeu ao serviço. Armazena-se no formato DATE yyyy-mm-dd do MySQL
4	hora	hora que cliente acedeu ao serviço (directório compartilhado)
5	id	identificador da sessão
6	ip	IP do cliente
7	directorio	directório compartilhado

Tabela 18 - Dados processados NFS

Através da figura 33 observam-se as posições dos dados retirados. O comando utilizado para limpar os logs é o da figura seguinte:

```
dados=(`cat $path | grep mountd | egrep -v Caught | awk '{print $x}' `)
```

Figura 34 - Comando extrair dados NFS

dados	posição(x)
mes	1
dia	2
hora	3
id	5
ip	10
directorio	12

Tabela 19 - Posição dos dados no log do NFS

O campo **id,** além deste comando, deve ser filtrado através do seguinte pipe de maneira a retirar-lhe o parêntesis recto:

```
id=(`cat $path | grep mountd | egrep -v Caught | awk '{print $5}' | cut -d"[" -f 2 | cut -d"]" -f 1 `)
```

Figura 35 - Pipe utilizado para obter id

O campo **ip** também é filtrado com outros comandos, como se pode observar na figura seguinte, para que apenas seja aproveitada a informação antes dos **":"**:

```
ip=(`cat $path | grep mountd | egrep -v Caught | awk '{print $10}' | cut -d":" -f 1`)
```

Figura 36 - Pipe utilizado para obter ip

NOTA:

O **mês** vem com a notação Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec. Com um **case** (ver formato no Anexo I) transforma-se para o número correspondente a cada mês (de 01 a 12).

Figura 37 - Notas NFS

A seguir observam-se os dados depois de processados:

```
7       2008-05-18      18:12:42        6547    192.168.11.103  /home/catarina/Desktop/capturas
7       2008-05-18      18:41:19        6547    192.168.11.103  /home/catarina/Desktop/capturas
```

Figura 38 - saída_nfs.txt

4.2.7 Sendmail

Sendmail é um Mail Transfer Agent (MTA) mais utilizado em todo mundo, porém também um dos mais complexos e "difíceis" de ser configurado.

Descendente do original ARPANET delivermail, suporta muitos tipos de transferências e envios, incluindo o popular SMTP. A versão original foi escrita por Eric Allman em 1980 na UC Berkley. A versão instalada neste trabalho foi a 8.13.8.

4.2.7.1 Ficheiros de logs

Os ficheiros de log registados no servidor são: **mail.log**, **mail.err**. São armazenados em **/var/log**. Também são criados pelo serviço de log de dados **Syslog**, descrito na RFC3164 [RFC3164 2008] e seguem o formato referenciado anteriormente no sub-capítulo 4.2.2.1. Apenas será analisado neste trabalho o **mail.log**.

A seguir encontra-se um trecho de um ficheiro de log:

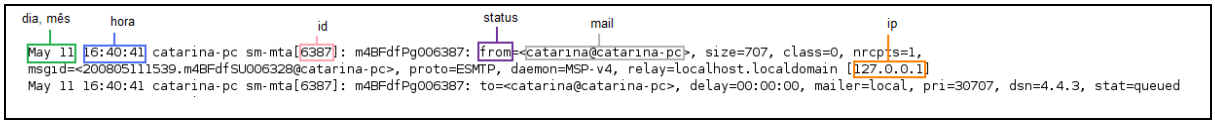


Figura 39 - mail.log

4.2.7.2 Sendmail Data Processing Script

Através do **script_mail.sh** fez-se o processamento dos dados registados no **mail.log**. No ficheiro de texto de saída, **saida_mail.txt**, os dados estam em colunas e separados por tabulações. Na tabela seguinte apresenta-se a descrição do **script_mail.sh**:

Coluna correspondente no ficheiro de saída	**Notação utilizada no script para os dados retirados do log**	**Significado**
1	vazia	Receberá o identificador da linha na base de dados
2	id_server	é o identificador do serviço numa

		determinada máquina. Recebe o valor do parâmetro de entrada, $1.
3	mes-dia. Não apresenta informação do ano (declara-se).	data que cliente acedeu ao serviço. Armazena-se no formato DATE yyyy-mm-dd do MySQL
4	hora	hora de envio ou recebimento do e-mail
5	id	identificação da sessão
6	ip	IP do remetente do e-mail
7	direction	pode ser TO ou FROM se o respectivo e-mail for enviado ou recebido respectivamente.
8	mail	e-mail

Tabela 20 - Dados processados Sendmail

Através da figura 39 observam-se as posições de cada elemento descrito na tabela acima. A seguir descrevem-se alguns dos comandos utilizados neste processamento:

```
mes=(`cat $path | grep @ | egrep -v queueing | awk '{print $1}' `)

dia=(`cat $path | grep @  | egrep -v queueing | awk '{print $2}' `)

hora=(`cat $path | grep @ | egrep -v queueing | awk '{print $3}' `)
```

O campo **ip** corresponde à informação do **relay** (ver figura 39). Porém, este aparece sempre em diferentes posições e nem em todas as linhas. Além disso, o comando **cut** apenas tem como delimitador um carácter ou uma posição. Como a palavra relay é uma string e esta informação não apresenta posição fixa, a solução foi substituir o **relay** por um **#** e depois cortar o que está depois do # e entre parêntesis recto, conforme comando seguinte:

```
ip=(`cat $path | grep @ | egrep -v queueing | sed 's/relay=/#/' | cut -d"#" -f 2 | cut -d"[" -f 2 | cut -d"]" -f 1  `)

id=(`cat $path | grep @ | egrep -v queueing | awk '{print $5}' | cut -d"[" -f 2 | cut -d"]" -f 1 `)

direction=(`cat $path | grep @ | egrep -v queueing | awk '{print $7}' | cut -d"=" -f 1`)
```

O campo **mail** por vezes aparece no formato **to=e-mail** e outras no **to=<e-mail>**. Logo, cortou-se com o seguinte comando:

```
mail=(`cat $path | grep @ | egrep -v queueing | awk '{print $7}' | cut -d"=" -f 2 | cut -d"<" -f 2 | cut -d"," -f 1 | cut -d">" -f 1 `)
```

NOTA:

O **mês** vem com a notação Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec. Com um **case** (ver formato no Anexo I) transforma-se para o número correspondente a cada mês (de 01 a 12).

Figura 40 - Notas Sendmail

A figura seguinte mostra o ficheiro de saída do **script_mail.sh**:

```
5       2008-05-11      16:40:41        6387    127.0.0.1       from    catarina@catarina-pc
5       2008-05-11      16:40:41        6387    ...     to      catarina@catarina-pc
```

Figura 41 - saida_mail.log

4.3 Notas ficheiros de log Linux

- Todos os ficheiros de log, dos serviços instalados nas máquinas Linux, quando alcançam um determinado tamanho mudam o seu nome X para X.0, e os dados mais recentes ficam registados num novo ficheiro com o nome X. Sucessivamente, este processo contínua e os ficheiros mais antigos são comprimidos. O intuito deste processo é diminuir o espaço ocupado pelos ficheiros de log no disco.
- Os hífenes surgem nos registos dos logs quando a saída de um determinado parâmetro não está disponível.
- No processamento feito através dos shell scripts teve-se em atenção que apenas devem ser armazenadas na base de dados as novas informações. Lê-se da base de dados através do comando SELECT a hora e data em que houve o último processamento. Compara-se com a data e hora actual, e apenas escreve-se nos ficheiros de saída a informação nova, para posteriormente, apenas esta informação ser escrita na base de dados. No fim actualiza-se na base de dados a hora e data do processamento.
- As informações escritas nos ficheiros de saída de todos os scripts dos serviços devem estar em colunas, em que cada coluna representa um campo da tabela, e separadas por tabulações. Apenas assim através do comando LOAD DATA INFILE é possível carregar os dados correctamente para a tabela respectiva na base de dados.
- Para os ficheiros de log que foram processados por partes, foi feita a ordenação dos dados por data e hora, através do comando descrito na figura 19, antes da passagem destes para a base de dados. Assim, há uma maior clareza na visualização dos dados através da interface *Web*.
- Nas figuras 14, 20, 24, 30, 32, 38 e 41, coluna 2, observa-se para cada serviço o identificador do mesmo utilizado neste trabalho.

4.4 Servidores instalados em máquina com sistema operativo Windows.

De forma a comparar o formato dos ficheiros de log em vários sistemas operativos Windows:

- Instalaram-se servidores Windows numa máquina virtual com Windows XP através do software Virtual Box.
- Instalou-se o Windows Server 2000.
- Instalou-se Windows Server 2003 R2 Enterprise Edition.

Os primeiros serviços descritos neste capítulo, *Web*, FTP, SMTP, pertencem ao Microsoft IIS. O Microsoft IIS é um conjunto integrado de serviços de rede para servidores em máquinas com sistemas operativos Windows. Apresenta interface gráfica, e com as diversas versões tem vindo a evoluir bastante a nível de segurança, o número de vulnerabilidades tem diminuído significativamente.

Neste trabalho, pôde-se verificar os ficheiros de log dos serviços em IIS5.0 (Windows Server 2000), IIS5.1 (Windows XP) e IIS6.0 (Windows Server 2003). Verificou-se que apesar da evolução sofrida por estes sistemas, os ficheiros de log destes serviços, informação que precisamos recolher neste trabalho, apresentam sempre os mesmos campos.

Os logs do sistema IIS suportam quatro formatos [IISFAQ2008]:

- W3C Extended Log File Format: campos são separados por espaços, tempo é referente ao meridiano de Greenwich, o ano é representado com quatro dígitos, e podemos seleccionar apenas os campos que queremos, formato é ASCII personalizável.
- Microsoft IIS Log Format: campos são separados por vírgulas, tempo tem referência local, o formato é ASCII fixo (não podemos alterar o número de campos), ano é representado com dois dígitos
- NCSA Common Log File Format: campos são separados por espaços, formato ASCII com número de campos fixo, ano é representado com quatro dígitos, não esta disponível para servidores FTP.
- ODBC Logging : formato fixo, disponível em base de dados como a Microsoft Access ou Microsoft SQL Server

Alem destes ficheiros de log, máquinas com sistema operativo Windows mantém registos de todas as tarefas e processos que ocorrem no computador. Estes eventos são registados pelo Event Log Service e dividem-se em três ficheiros:

AppEvent.evt - Application: são gerados pelas aplicações.

SecEvent.evt - Security: incluem instâncias de ligação e saída (logging e logoff).

SysEvent.evt - System: gerados pelo sistema operativo e detalham instâncias relacionadas com drivers, dispositivos de hardware e aos processos gerados pelo sistema operacional.

Dois dos serviços estudados no âmbito deste trabalho, o terminal e file server, registam os seus eventos no SecEvent.evt, como descrito nos sub-capítulos 4.3.5 e 4.3.6.

A instalação e configuração dos serviços Windows encontram-se no Anexo III.

4.4.1 IISWeb

É um completo servidor *Web* que permite um rápido e fácil desenvolvimento de *Web* sites e aplicações. É possível gerar páginas HTML dinâmicas, usando tecnologia proprietária ASP (Active Server Pages) ou outras tecnologias através de adição de módulos de terceiros. Neste trabalho, escuta no porto padrão, o 80.

4.4.1.1 Ficheiros de logs

O formato escolhido para os logs deste serviço foi o W3C Extended Log File Format e o critério de armazenamento dos dados foi diariamente, logo, cada nome é armazenado com a notação ex**yymmdd**.log em que yy é o ano, mm é o mês e dd é o dia. Por defeito, ficam localizados em **%windir%\System32\LogFiles\W3SVC1**.

O conteúdo destes ficheiros de log e o que significa cada campo esta referenciado na tabela 22. Para uma melhor percepção, estes campos apresentam as seguintes notações:

Prefixo	**Significado**
s-	Acções do servidor
c-	Acções do cliente
cs-	Acções de cliente para servidor
sc-	Acções de servidor para cliente

Tabela 21 - Notações utilizadas nos campos

Campo	**Descrição**
date	data
time	hora
c-ip	endereço IP do cliente
cs-username	nome do usuário que utilizou o serviço. Se for anónimo ou não for possível a identificação é substituído por hífen.
s-sitename	nome do serviço
s-computername	nome da máquina servidor

s-ip	endereço ip do servidor
s-port	porto que o serviço escuta
cs-method	método
cs-uri-stem	tronco URI
cs-uri-query	consulta URI
sc-status	status do servidor
sc-substatus	sub-status do servidor
sc-win32-status	status da acção
sc-bytes	número de bytes enviados pelo servidor
cs-bytes	número de bytes recebidos pelo servidor
time-taken	tempo para completar a transacção em segundos
cs-version	versão
cs-host	nome do host
cs(User-Agent)	browser utilizado no cliente
cs(Cookie)	conteúdo do cookie, enviado ou recebido
cs(Referer)	página *Web* visitada anteriormente pelo usuário

Tabela 22 - Campos W3C Extended Log

Na figura seguinte observa-se um trecho de um ficheiro de log deste serviço:

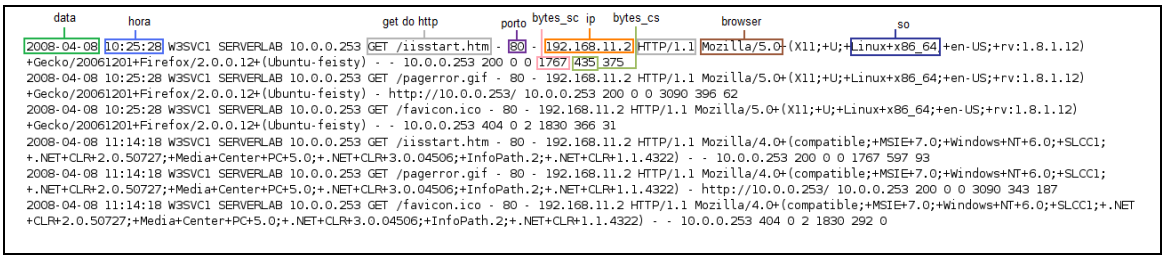


Figura 42 - Ficheiro de log IISWeb

4.4.1.2 IISWeb Data Processing Script

Através do **script_win_web.sh** fez-se o processamento dos dados registados nos ficheiros de log. Para extrair estes dados fez-se uma função em shell script:

```
function indice()
{
for (( i=0 ; i<=22 ; i++ ))
do
word=(`cat $path | grep Fields | cut -d" " -f $i `)
      if [ "$word" = "$1" ]; then
            let r=$i-1
      fi
done
}
```

Figura 43 - Função indice

Esta função foi utilizada pois, os campos nos ficheiros de log W3C, como referido anteriormente, não são de tamanho fixo, e sim é o administrador do servidor que escolhe quais informações quer que fiquem registadas. Assim, os campos não têm sempre a mesma posição relativamente à coluna que estão. Deve-se então procurar as posições correspondentes a cada campo desejado. A função procura na linha **Fields** (linha que apresenta o nome e sequência dos campos), percorre palavra por palavra (separadas por espaço) até encontrar a palavra igual a desejada (parâmetro de entrada) e retorna a posição **($r)**.

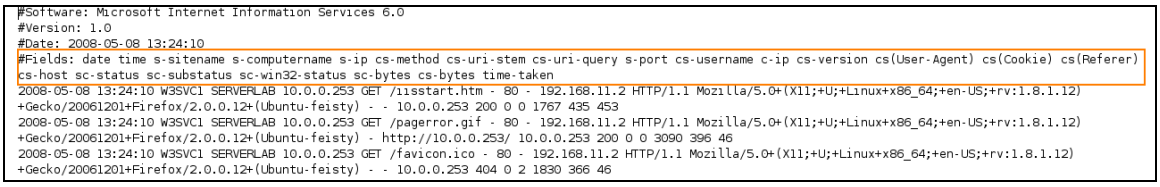


Figura 44 - Fields no ficheiro de log

No ficheiro de texto de saída, **saida_win_web.txt**, os dados devem estar em colunas e separados por tabulações para que a base de dados possa interpretar e carregar os dados através do comando LOAD DATA INFILE de forma correcta. Na tabela seguinte apresenta-se a descrição do **script_win_web.sh**:

Coluna correspondente no ficheiro de saída	**Notação utilizada no script para os dados retirados do log**	**Significado**	**Field**
1	vazia	Receberá o identificador da linha na base de dados	-
2	id_server	é o identificador do serviço numa determinada máquina. Recebe o valor do parâmetro de entrada, $1.	-
3	data	data que o cliente acedeu ao serviço	date
4	hora	hora que cliente acedeu ao serviço	time
5	ip	IP do cliente	c-ip
6	get do http	pedido feito pelo cliente.	cs-method cs-uri-stem cs-version
7	browser	browser utilizado pelos cliente	cs(User-Agent)
8	so	sistema operativo na máquina do cliente	cs(User-Agent)
9	bytes_cs	número de bytes transmitidos do cliente para o servidor	cs-bytes
10	bytes_sc	número de bytes transmitidos	sc-bytes

		do servidor para o cliente	
11	porto	porto que o servidor escuta	s-port

Tabela 23 - Dados processados IISWeb

Todos estes dados são então extraídos com o comando da figura seguinte utilizando as **fields** que estão na quarta coluna da tabela anterior:

```
indice field
notação_script=(`cat $path | egrep -v "^#" |cut -d" " -f $r `)
```

Figura 45 - Utilização da função indice

No processamento dos dados só são analisadas as linhas sem comentários através do comando shell **egrep**.

Os campos **browser** e **so** como se pode ver na tabela 23, partilham o mesmo campo, logo ainda precisam utilizar os seguintes pipes:

browser=(`cat $path | egrep -v "^#" | cut -d" " -f $r | cut -d"+" -f 1 `)

so= cat $path | egrep -v "^#" | cut -d" " -f $r | cut -d";" -f 3 | sed '/\+/s// /g'

Na figura seguinte está representada o ficheiro da figura 42, após o processamento:

```
8   2008-04-08  10:25:28  192.168.11.2  GET /iisstart.htm HTTP/1.1  Mozilla/5.0  Linux x86_64    1767  435  80
8   2008-04-08  10:25:28  192.168.11.2  GET /pagerror.gif HTTP/1.1  Mozilla/5.0  Linux x86_64    3090  396  80
8   2008-04-08  10:25:28  192.168.11.2  GET /favicon.ico HTTP/1.1   Mozilla/5.0  Linux x86_64    1830  366  80
8   2008-04-08  11:14:18  192.168.11.2  GET /iisstart.htm HTTP/1.1  Mozilla/4.0  Windows NT 6.0  1767  597  80
8   2008-04-08  11:14:18  192.168.11.2  GET /pagerror.gif HTTP/1.1  Mozilla/4.0  Windows NT 6.0  3090  343  80
8   2008-04-08  11:14:18  192.168.11.2  GET /favicon.ico HTTP/1.1   Mozilla/4.0  Windows NT 6.0  1830  292  80
```

Figura 46 - saída_win_web.txt

4.4.2 IISFTP

É um serviço de acesso de utilizadores a um disco ou directório através do protocolo de transporte FTP (File Transfer Protocol). Escuta no porto 21.

4.4.2.1 Ficheiros de logs

O formato escolhido para os logs deste serviço foi o W3C Extended Log File Format e o critério de armazenamento dos dados foi diariamente. Assim, cada nome é armazenado com a notação ex**yymmdd**.log em que yy é o ano, mm é o mês e dd é o dia. Estes, por defeito, ficam em **%windir%\System32\LogFiles\MSFTPSVC1.**

Apresentam os mesmos campos do serviço *Web* listados na tabela 22.

Na figura seguinte apresenta-se um trecho do ficheiro de log deste serviço:

```
2008-05-06 19:32:21 192.168.11.2 mozilla@example.com MSFTPSVC1 SERVERLAB 10.0.0.253 21 [2]PASS mozilla@example.com - 230 0 0 0 0 FTP - - - -
2008-05-06 19:32:21 192.168.11.2 mozilla@example.com MSFTPSVC1 SERVERLAB 10.0.0.253 21 [2]sent / - 550 2 0 0 0 FTP - - - -
2008-05-06 19:32:21 192.168.11.2 mozilla@example.com MSFTPSVC1 SERVERLAB 10.0.0.253 21 [2]sent / - 426 2 0 0 0 FTP - - - -
2008-05-06 19:32:21 192.168.11.2 mozilla@example.com MSFTPSVC1 SERVERLAB 10.0.0.253 21 [2]CWD / - 250 0 0 0 0 FTP - - - -
2008-05-06 19:32:23 192.168.11.2 mozilla@example.com MSFTPSVC1 SERVERLAB 10.0.0.253 21 [2]sent / - 550 2 0 0 0 FTP - - - -
2008-05-06 19:32:23 192.168.11.2 mozilla@example.com MSFTPSVC1 SERVERLAB 10.0.0.253 21 [2]sent / - 426 2 0 0 0 FTP - - - -
2008-05-06 19:32:23 192.168.11.2 mozilla@example.com MSFTPSVC1 SERVERLAB 10.0.0.253 21 [2]CWD / - 250 0 0 0 0 FTP - - - -
2008-05-06 19:32:25 192.168.11.2 mozilla@example.com MSFTPSVC1 SERVERLAB 10.0.0.253 21 [2]sent /DhcpSrvLog-Mon.log - 226 0 2462 0 0 FTP - - - -
2008-05-06 19:35:11 192.168.11.2 mozilla@example.com MSFTPSVC1 SERVERLAB 10.0.0.253 21 [2]closed - - 421 121 0 0 165703 FTP - - - -
data  hora  ip  browser  porto  status  documento  bytes_sc  bytes_cs
```

Figura 47 - Ficheiro de log IISFTP

4.4.2.2 IISFTP Data Processing Script

Através do **script_win_ftp.sh** fez-se o processamento dos dados registados nos ficheiros de log.

As linhas do ficheiro de log apresentavam diferentes informações, por isso para um melhor processamento foi dividido em duas partes (linhas com palavra **sent** e as outras linhas). No fim, no ficheiro de texto de saída **saida_win_ftp.txt**, os dados estão mais uma vez em colunas e separados por tabulações e há uma ordenação prévia dos dados (com o comando da figura 19). Na tabela seguinte apresenta-se a descrição do **script_win_ftp.sh**:

Coluna correspondente no ficheiro de saída	**Notação utilizada no script para os dados retirados do log**	**Significado**	**Field**
1	vazia	Receberá o identificador da linha na base de dados	-
2	id_server	é o identificador do serviço numa determinada máquina. Recebe o valor do parâmetro de entrada, $1.	-
3	data	data de login do cliente. Fica no formato DATE (yyyy-mm-dd) do MySQL	date
4	hora	hora que o cliente acedeu ao serviço	time
5	ip	IP do cliente	c-ip
6	browser	browser utilizado pelos clientes	cs-username
7	status	status do servidor	cs-method
8	documento	documento que foi feita a transferência	cs-uri-stem. Apenas nas linhas com a palavra **sent**.
9	bytes_sc	numero de bytes do ficheiro numa transferência servidor cliente	sc-bytes
10	bytes_cs	numero de bytes do ficheiro numa transferência cliente servidor	cs-bytes
11	porto	porto que o servidor escuta	s-port

Tabela 24 - Dados processados IISftp

Foi usada também a função **indice** descrita na figura 43, e da mesma forma foram processados os dados com o comando da figura 45 utilizando as **fields** que estão na quarta coluna da tabela anterior. Porém, os campos **browser** e **status** ainda foram filtrados da seguinte forma:

browser=(`cat $path_auxiliar | cut -d" " -f $r | cut -d"@" -f 1 `)

status=(`cat $path_auxiliar | cut -d" " -f $r | cut -d"]" -f 2`)

Depois de processados, os dados da figura 47 apresentam-se no ficheiro **saida_win_ftp.txt** da seguinte forma:

9	2008-05-06	19:32:21	192.168.11.2	mozilla CWD		0	0	21	
9	2008-05-06	19:32:21	192.168.11.2	mozilla PASS		0	0	21	
9	2008-05-06	19:32:21	192.168.11.2	mozilla sent	/	0	0	21	
9	2008-05-06	19:32:21	192.168.11.2	mozilla sent	/	0	0	21	
9	2008-05-06	19:32:23	192.168.11.2	mozilla CWD		0	0	21	
9	2008-05-06	19:32:23	192.168.11.2	mozilla sent	/	0	0	21	
9	2008-05-06	19:32:23	192.168.11.2	mozilla sent	/	0	0	21	
9	2008-05-06	19:32:25	192.168.11.2	mozilla sent	/DhcpSrvLog-Mon.log	2462	0	21	
9	2008-05-06	19:35:11	192.168.11.2	mozilla closed		0	0	21	

Figura 48 - saída_win_ftp.txt

4.4.3 DHCP

A função de um servidor Dynamic Host Configuration Protocol (DHCP), definido anteriormente neste trabalho – sub-capítulo 4.2.3 - e é sumarizada através das suas siglas:

- **D**ynamic: significa que o endereço dos clientes pode mudar
- **H**ost: indica que é um sistema para clientes
- **C**onfiguration: configurável no servidor
- **P**rotocol: segue um conjunto de regras (protocolo)

4.4.3.1 Ficheiros de logs

Estão localizados em **%windir%\System32\Dhcp** e são armazenados por dia da semana com a notação: **DhcpSrvLog-x.log** em que **x** pode ser **Mon**, **Tue**, **Wed**, **Thu**, **Fri**, **Sat**, **Sun**, e são substituídos de semana em semana. Nos ficheiros de log os campos são separados por vírgulas. Os campos registados encontram-se na seguinte tabela:

Campo	**Descrição**
ID	identificador do evento no servidor
Date	a data do evento
Time	a hora do evento
Description	a descrição do evento no servidor. Pode ser dentre outros *assign, release, stopped*
IP Address	o IP do cliente dhcp
Host Name	o nome do cliente
MAC Address	endereço MAC da interface do cliente

Tabela 25 - Campos DHCP

Na figura seguinte apresenta-se um trecho do ficheiro de log:

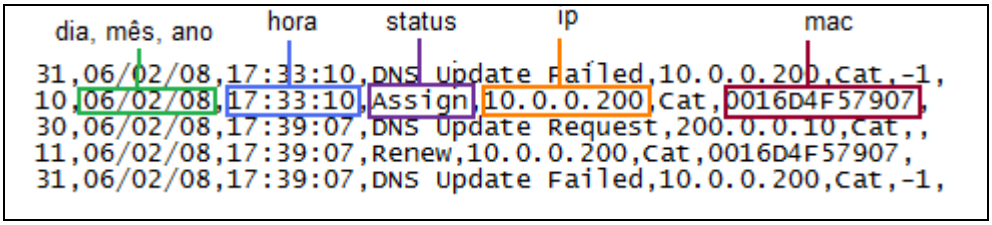


Figura 49 - Ficheiro de log DHCP

4.4.3.2 DHCP Data Processing Script

Através do **script_dhcp.sh** fez-se o processamento dos dados registados nos ficheiros de log. Apenas se armazenaram os dados correspondentes às linhas dos processos **Assign** e **Renew**. No ficheiro de texto de saída, **saida_dhcp.txt**, os dados devem estar em colunas e separados por tabulações para que a base de dados possa interpretar e carregar correctamente os dados através do comando LOAD DATA INFILE. Na tabela seguinte apresenta-se a descrição do **script_dhcp.sh**:

Coluna correspondente no ficheiro de saída	**Notação utilizada no script para os dados retirados do log**	**Significado**
1	vazia	Receberá o identificador da linha na base de dados
2	id_server	é o identificador do serviço numa determinada máquina. Recebe o valor do parâmetro de entrada, $1.
3	ano-mes-dia	data que cliente acedeu ao serviço. Armazena-se no formato DATE yyyy-mm-dd do MySQL
4	hora	hora que cliente acedeu ao serviço
5	status	pode ser Assign ou Renew.
6	ip	IP do cliente
7	mac	MAC da interface do cliente

Tabela 26 - Dados processados DHCP

Através da figura 49 observam-se as posições dos elementos e através dos seguintes comandos foi feito o processamento dos dados:

```
ano=(`cat $path | cut -d"," -f 2 | cut -d"/" -f 3`)

mes=(`cat $path | cut -d"," -f 2 | cut -d"/" -f 1`)

dia=(`cat $path | cut -d"/" -f 2 `)

hora=(`cat $path | cut -d"," -f 3`)

status=(`cat $path1 | cut -d"," -f 4`)

ip=(`cat $path1 | cut -d"," -f 5`)
```

mac=(`cat $path1 | cut -d"," -f 7`)

Depois de toda a informação processada, o ficheiro de saída fica como na seguinte figura:

```
10      2008-06-02      17:33:10        Assign  10.0.0.200      0016D4F57907
10      2008-06-02      17:39:07        Renew   10.0.0.200      0016D4F57907
```

Figura 50 - saída_win_dhcp.txt

4.4.4 IISSMTP

O SMTP (Simple Mail Transfer Protocol) é um protocolo do TCP/IP que permite transmitir mensagens de um computador a outro em rede. O IIS inclui um servidor de SMTP com todas as funcionalidades que podem ser utilizados para receber e reencaminhar mensagens de correio electrónico para outros servidores SMTP na sua rede ou para servidores na Internet. A função de reencaminhamento é útil para os clientes da rede interna que poderão ter de reencaminhar correio para outros servidores SMTP e para o IIS, programas que necessitam de aceder a um servidor de SMTP para reencaminhar correio. O servidor SMTP inclui melhorias para além das funções básicas do protocolo. Há opções disponíveis que fornecem controlo do encaminhamento e envio das mensagens e que fornecem comunicações seguras. Por defeito utiliza o porto 25.

4.4.4.1 Ficheiros de logs

O formato para os logs escolhido foi o W3C Extended Log File Format e o critério de armazenamento dos dados foi diariamente, logo, cada nome é armazenado com a notação ex**yymmdd**.log em que yy é o ano, mm é o mês e dd é o dia.

Apresentam os mesmos campos do servidor *Web* listados na tabela 22. Na figura seguinte é apresentado um trecho do ficheiro de log:

```
                data     hora                                                    ip                                direction       mail                                  bytes_sc bytes_sc
2008-02-19 11:42:47 192.168.0.2 Cat SMTPSVC1 SERVERLAB 192.168.0.1 0 EHLO - +Cat 250 0 185 8 47 SMTP - - - -
2008-02-19 11:42:47 192.168.0.2 Cat SMTPSVC1 SERVERLAB 192.168.0.1 0 MAIL - +FROM:+<sermail@serverit.com> 250 0 45 33 31 SMTP - - - -
2008-02-19 11:42:47 192.168.0.2 Cat SMTPSVC1 SERVERLAB 192.168.0.1 0 RCPT - +TO:+<sermail@serverit.com> 250 0 33 31 0 SMTP - - - -
2008-02-19 11:42:47 192.168.0.2 Cat SMTPSVC1 SERVERLAB 192.168.0.1 0 DATA - +<000001c872ec$7fd34660$7f79d320$@com> 250 0 121 2797 125 SMTP - - - -
2008-02-19 11:42:49 192.168.0.2 Cat SMTPSVC1 SERVERLAB 192.168.0.1 0 QUIT - Cat 240 2781 58 4 0 SMTP - - - -
```

Figura 51 - Ficheiro de log IISSMTP

4.4.4.2 IISSMTP Data Processing Script

Através do **script_win_mail.sh** fez-se o processamento dos dados registados nos ficheiros de log. Foram utilizadas no processamento as linhas com as palavras **FROM** e **TO**. No fim, no ficheiro de texto de saída **saida_win_mail.txt**, os dados estão mais uma vez em colunas e separados por tabulações.

Na tabela seguinte apresenta-se a descrição do **script_win_mail.sh**:

Coluna correspondente no ficheiro de saída	Notação utilizada no script para os dados retirados do log	Significado	Field
1	vazia	Receberá o identificador da linha na base de dados	-
2	id_server	é o identificador do serviço numa determinada máquina. Recebe o valor do parâmetro de entrada, $1.	-
3	data	data de envio ou recebimento do e-mail	date
4	hora	hora de envio ou recebimento do e-mail	time
5	ip	IP do cliente	c-ip
6	direction	pode ser FROM ou TO	cs-uri-query
7	mail	e-mail de origem ou de destino (se o direction é FROM ou TO)	cs-uri-query
8	bytes_sc	numero de bytes do ficheiro numa transferência servidor cliente	sc-bytes
9	bytes_cs	numero de bytes do ficheiro numa transferência cliente servidor	cs-bytes

Tabela 27 - Dados processados IISSMTP

O ficheiro de log segue também o mesmo formato dos outros serviços IIS (W3C Extended Format File). Deste modo pode-se utilizar também a função **índice,** descrita na figura 43, e o comando da figura 45 utilizando as **fields** que estão na quarta coluna da tabela anterior. Além de se utilizar este método, os campos **direction** e **mail** por fazerem parte do mesmo **field** foram ainda filtrados com os comandos apresentados a seguir pois, o primeiro está entre os caracteres **":"** e **"+"**, e o segundo está entre os símbolos **"<"** e **">"**.

```
direction=(`cat $path1 | egrep -v "^#" | cut -d" " -f $r | cut -d":" -f1 | cut -d"+" -f2 `)

mail=(`cat $path1 | egrep -v "^#" | cut -d" " -f $r | cut -d":" -f2 | cut -d"<" -f2 | cut -d">" -f1 `)
```

A seguir encontra-se o ficheiro da figura 51 depois de processado:

```
11      2008-02-19      11:42:47        192.168.0.2     FROM    sermail@serverit.com    45      33
11      2008-02-19      11:42:47        192.168.0.2     TO      sermail@serverit.com    33      31
```

Figura 52 - Saída IISSMTP

4.4.5 Terminal

É um serviço que permite o acesso ao servidor remotamente trabalhando-se em ambiente gráfico. Utiliza o protocolo RDP. No WinServer2003 a versão deste protocolo é a 5.2 apresentando melhorias em relação a versão 5.1 do WinServer2000, porém não é do âmbito deste trabalho cita-las.

4.4.5.1 Ficheiros de log

Este serviço não apresenta um ficheiro de log específico. Os eventos são registados no log de eventos de segurança geral do sistema operativo, o **SecEvent.evt**. Este fica no directório **%system$user%\System32\config**, tem restrições de acesso, e como pode-se observar pelo nome apresenta um formato personalizado. Para a sua visualização é possível a mudança de formato para texto (.txt) através do programa Event Viewer (já vem instalado no WinServer e noutros sistemas operativos Windows) ou através do GrokEVT [GrokEVT2008], que é uma colecção de scripts que interpreta os dados através de uma base de dados. Neste trabalho utiliza-se o ficheiro em formato de texto (.txt).

Neste ficheiro de log são registados os eventos com e sem êxito. Os eventos são separados por blocos e cada um contém três partes: cabeçalho, descrição e uma secção de dados binários.

Observou-se que os eventos correspondentes a este serviço têm a seguinte informação [Microsoft2008]:

	Cabeçalho	Descrição
	Identificação do evento	**Logon type**
Logon	528	10
Logoff	683	-

Tabela 28 - Identificadores de login/logoff no servidor Terminal

Na figura seguinte apresenta-se um pequeno trecho do log correspondente a este serviço:

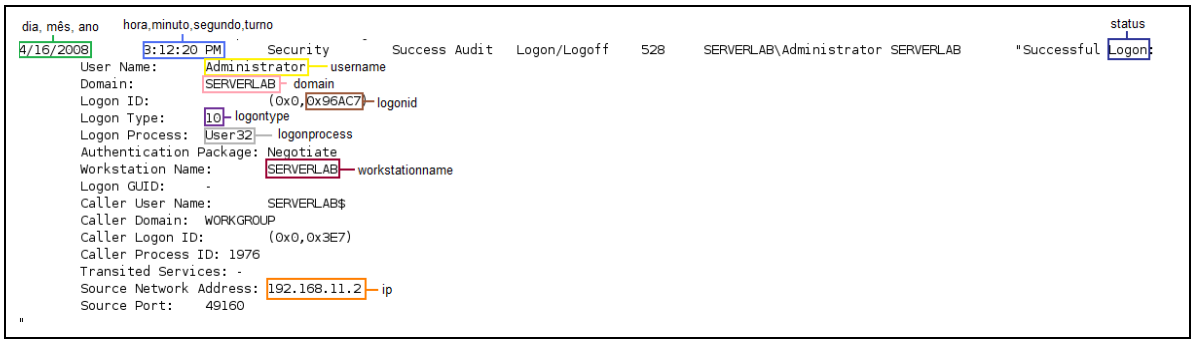


Figura 53 - Ficheiro de log - logon Terminal

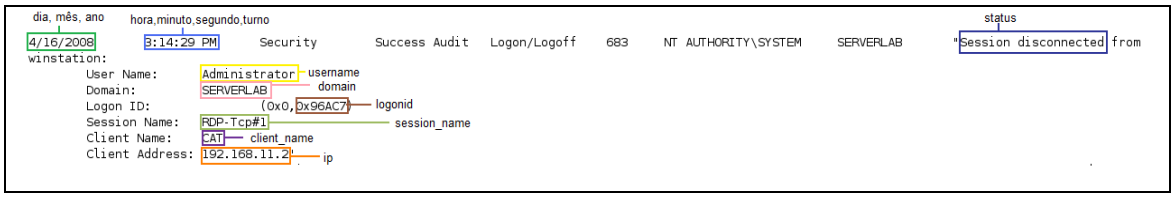


Figura 54 - Ficheiro de log - logoff Terminal

4.4.5.2 Terminal Data Processing Script

Como se pode observar nas figuras anteriores 53 e 54, o conteúdo do ficheiro **SecEvent.txt**, ao contrário dos outros casos estudados ate aqui, não segue um padrão comum entre linhas. É constituído por blocos cujo conteúdo e número de linhas são diferentes, e não há nenhum carácter separador especial entre cada bloco. Assim, a solução foi processar linha a linha, através do **script_terminal.sh**. Se fossem encontrados os requisitos descritos na tabela 28 analisavam-se as linhas que se seguem a linha do cabeçalho e escrevia-se cada uma para um ficheiro auxiliar. Destes ficheiros auxiliares são retirados os dados descritos na seguinte tabela através do **script_terminal.sh**. Observam-se os comandos utilizados neste script de processamento após a tabela seguinte.

Coluna correspondente no ficheiro de saída	**Notação utilizada no script para os dados retirados do log**	**Significado**
1	vazia	Receberá o identificador da linha na base de dados
2	id_server	é o identificador do serviço numa determinada máquina. Recebe o valor do parâmetro de entrada, $1.
3	ano-mes-dia	data que cliente acedeu ao serviço. Armazena-se no formato DATE yyyy-mm-dd do MySQL
4	hora, minuto, segundo e turno (auxiliar)	hora que cliente acedeu remotamente
5	status	se cliente fez logon ou logoff
6	logonid	identificador da ligação
7	logontype	tipo de logon
8	logonprocess	nome do processo de logon
9	ip	IP do cliente
10	username	username utilizado pelo cliente (deve haver uma permissão de acesso deste utilizador na máquina servidora)
11	client_name	nome do cliente
12	workstationname	nome do computador servidor
13	domain	nome do servidor
14	session_name	nome da sessão RDP

Tabela 29 - Dados processados Terminal

Comandos:

```
ano=(`cat $path_auxiliar1 | awk '{print $1}' | cut -d"/" -f 3 `)
```

```
mes=(`cat $path_auxiliar1 | awk '{print $1}'| cut -d"/" -f 1 `)
dia=(`cat $path_auxiliar1 | awk '{print $1}' | cut -d"/" -f 2 `)
hora=(`cat $path_auxiliar1 | awk '{print $2}'| cut -d":" -f 1 `)
min=(`cat $path_auxiliar1 | awk '{print $2}'| cut -d":" -f 2 `)
sec=(`cat $path_auxiliar1 | awk '{print $2}'| cut -d":" -f 3 `)
turno=(`cat $path_auxiliar1 | awk '{print $3}' `)
status=(`cat $path_auxiliar1 | awk '{print $12 $13}'| cut -d":" -f 1| cut -d"\"" -f 2`)
logonid=(`cat $path_auxiliar4 | awk '{print $3}'| cut -d"," -f 2 | cut -d")" -f 1`)
logontype=(`cat $path_auxiliar5 | awk '{print $3}"`)
logonprocess=(`cat $path_auxiliar6 | awk '{print $3}"`)
ip=(`cat $path_auxiliar8 | cut -d":" -f 2 | cut -d"\"" -f 1 `)
username=(`cat $path_auxiliar2 | cut -d" " -f 3 `)
client_name=(`cat $path_auxiliar10 | awk '{print $3}"`)
workstationname=(`cat $path_auxiliar7 | awk '{print $3}"`)
domain=(`cat $path_auxiliar3 | cut -d" " -f 2 `)
session_name=(`cat $path_auxiliar9 | awk '{print $3}"`)
```

Depois de todo o processamento descrito, os dados no ficheiro de saída **saida_terminal.txt** apresentam a seguinte forma:

```
13    2008-4-16    15:12:20    Logon   0x96AC7 10    User32  192.168.11.2   Administrator   -     SERVERLAB     SERVERLAB     -
13    2008-4-16    15:14:29    Logoff  0x96AC7 -     -       192.168.11.2   Administrator   CAT   -     SERVERLAB     RDP-Tcp#1
```

Figura 55 - saida_terminal.txt

4.4.6 File

Um file server é um servidor responsável pelo armazenamento e gerência dos ficheiros em uma posição central. Os clientes da rede podem ter acesso a estes ficheiros e modifica-los sem ter de fisicamente transferi-lo para o seu computador.

4.4.6.1 Ficheiros de logs

Da mesma forma que o serviço terminal, este não apresenta um ficheiro de log específico. Os eventos também são registados no log de eventos de segurança geral do sistema operativo, o **SecEvent.evt**. Todo o processo descrito no serviço terminal é realizado aqui também. Porém, observou-se que os eventos correspondentes a este serviço têm a seguinte informação:

	Cabeçalho	Descrição	
	Identificação do evento	**Logon process**	**Logon Type**
Logon	540	NtLmSsp	-
Logoff	538	-	3

Tabela 30 - Identificadores de login/logoff no servidor File

Na figura seguinte apresenta-se um pequeno trecho do ficheiro de log, correspondente a este serviço, em formato .txt:

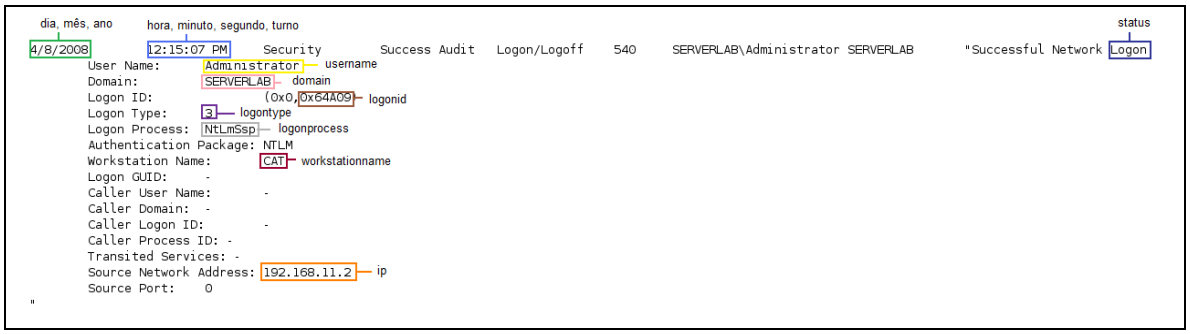


Figura 56 - Ficheiro de log - logon File

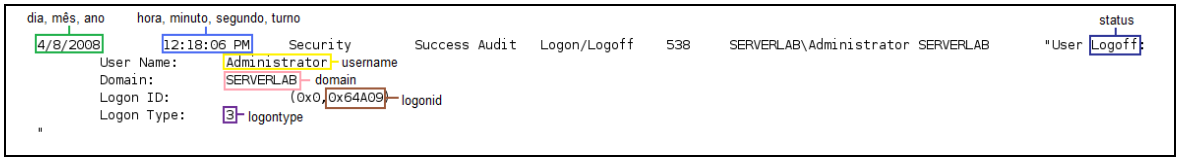


Figura 57 - Ficheiro de log - logoff File

4.4.6.2 File Data Processing Script

Pelos mesmos motivos referidos no terminal server e observando as figuras anteriores 56 e 57, processou-se o ficheiro de linha em linha através do **script_file.sh**, analisando-se as linhas correspondentes as informações descritas na tabela 30. Se fossem encontrados os requisitos descritos nesta tabela analisavam-se as linhas que se seguem a linha do cabeçalho e escrevia-se cada uma para um ficheiro auxiliar. Destes ficheiros auxiliares são retirados os dados descritos na seguinte tabela. Estes dados são escritos em colunas e separados por tabulações no ficheiro de texto de saída, **saida_file.txt**.

Coluna correspondente no ficheiro de saída	**Notação utilizada no script para os dados retirados do log**	**Significado**
1	vazia	Receberá o identificador da linha na base de dados
2	id_server	é o identificador do serviço numa determinada máquina. Recebe o valor do parâmetro de entrada, $1.
3	ano-mes-dia	data que cliente acedeu ao serviço e armazena-se no formato DATE

		yyyy-mm-dd do MySQL
4	hora, minuto, segundo e turno (auxiliar)	hora que cliente acedeu ao serviço
5	status	se cliente fez logon ou logoff
6	logonid	identificador da ligação
7	logontype	tipo de *logon*
8	logonprocess	nome do processo de *logon*
9	ip	IP do cliente
10	username	username utilizado pelo cliente (deve haver uma permissão de acesso deste utilizador na máquina servidora)
11	workstationname	nome do computador servidor
12	domain	nome do servidor

Tabela 31 - Dados processados File

Observam-se os comandos utilizados no **script_file.sh:**

```
ano=(`cat $path_auxiliar1 | awk '{print $1}' | cut -d"/" -f 3 `)

mes=(`cat $path_auxiliar1 | awk '{print $1}'| cut -d"/" -f 1 `)

dia=(`cat $path_auxiliar1 | awk '{print $1}' | cut -d"/" -f 2 `)

hora=(`cat $path_auxiliar1 | awk '{print $2}'| cut -d":" -f 1 `)

min=(`cat $path_auxiliar1 | awk '{print $2}'| cut -d":" -f 2 `)

sec=(`cat $path_auxiliar1 | awk '{print $2}'| cut -d":" -f 3 `)

turno=(`cat $path_auxiliar1 | awk '{print $3}' `)

status=(`cat $path_auxiliar1 | awk '{print $12 $13 $14 $15}'| cut -d":" -f 1 | cut -d"\"" -f 2`)

logonid=(`cat $path_auxiliar4 | awk '{print $3}' | cut -d"," -f 2 | cut -d")" -f 1 `)

logontype=(`cat $path_auxiliar5 | awk '{print $3}' `)

logonprocess=(`cat $path_auxiliar6 | awk '{print $3}' `)

ip=(`cat $path_auxiliar8 | awk '{print $4}' `)

username=(`cat $path_auxiliar2 | cut -d" " -f 3 `)

workstationname=(`cat $path_auxiliar7 | cut -d" " -f 3 `)

domain=(`cat $path_auxiliar3 | awk '{print $2}' `)
```

Depois de todo o processamento descrito, os dados no ficheiro de saída **saida_file.txt** ficam da seguinte forma:

```
12      2008-4-8        12:15:07        Logon   0x64A09 3       NtLmSsp -       Administrator   CAT     SERVERLAB
12      2008-4-8        12:18:06        Logoff  0x64A09 3       -       192.168.11.2    Administrator   -       SERVERLAB
```

Figura 58 - saida_file.txt

4.5 Notas ficheiros de log Windows

- Os hífenes surgem nos registos dos logs quando a saída de um determinado parâmetro não está disponível.
- Nos casos em que o nome do ficheiro de log é a data, no processamento dos dados pelos scripts lê-se da base de dados, através do comando SELECT, a hora e data em que houve o último processamento. Compara-se com a data e hora actual e processa-se apenas os ficheiros cuja data é igual ou superior a data do ultimo processamento, e nos casos que a data é a mesma, apenas escreve-se no ficheiro de saída as informações cuja hora é maior que a hora do último processamento.
- Nos casos em que o nome do ficheiro é sempre o mesmo, no processamento feito teve-se em atenção que apenas devem ser armazenadas na base de dados as novas informações. Lê-se da base de dados através do comando SELECT a hora e data em que houve o último processamento. Compara-se com a data e hora actual, e apenas escreve-se nos ficheiros de saída a informação nova, para posteriormente, apenas esta informação ser escrita na base de dados. No fim actualiza-se na base de dados a hora e data do processamento.
- Para os ficheiros de log que foram processados por partes, foi feita a ordenação dos dados por data e hora, através do comando descrito na figura 19, antes da passagem destes para a base de dados. Assim, há uma maior clareza na visualização dos dados através da interface *Web*.
- As informações escritas nos ficheiros de saída de todos os scripts dos serviços devem estar em colunas, em que cada coluna representa um campo da tabela, e separadas por tabulações. Apenas assim através do comando LOAD DATA INFILE é possível carregar os dados correctamente para a tabela respectiva na base de dados.
- Nas figuras 46, 48, 50, 52, 55 e 58, coluna 2, observa-se para cada serviço o identificador do mesmo utilizado neste trabalho.

4.6 Síntese

Neste capítulo foram apresentados todos os serviços e seus respectivos ficheiros de log. Foram escolhidos para estudo os principais serviços utilizados nas redes, com um total de treze, sendo sete serviços em máquinas Linux e seis em máquina Windows. Processaram-se os respectivos ficheiros de log através de shell scripts, e todo este processamento realizado teve como objectivo tornar mais clara e perceptível toda a informação registada nos logs. Toda esta informação pode vir a ser correlacionada para futura análise forense.

5 Monitorização dos equipamentos de rede

5.1 Introdução

De maneira a efectivamente monitorizar uma rede, é de essencial importância controlar também os seus equipamentos de rede. Saber o estado de cada interface, o tráfego que a atravessa, os portos ocupados, entre outras informações, sendo assim fundamental na detecção de actividades maliciosas. Para obter as informações dos equipamentos utilizou-se o protocolo SNMP.

5.2 Protocolo SNMP

Simple Network Management Protocol (SNMP) é um protocolo de gestão de redes. Permite que dispositivos de rede que utilizam o protocolo IP (Internet Protocol) sejam geridos remotamente. Consiste numa relação entre os seguintes componentes [Domingues2004]:

- Network Management System (NMS): aplicação que monitoriza e controla os *managed devices* realizando todo o processamento da informação de gestão recolhida. No âmbito deste trabalho, teremos um NMS, na máquina do administrador da rede.
- Agente SNMP: módulo de software de gestão de rede existente num *managed device* que compila informação sobre a entidade gerida onde reside, podendo ser um programa separado (um *daemon* num servidor Linux) ou incorporado (por exemplo, no Cisco IOS). Traduz informação armazenada na MIB para linguagem compreendida pelo SNMP.
- Managed devices (MD): são equipamentos de rede que contém um agente SNMP. Correlacionam e armazenam informação de gestão, de forma a ser disponibilizada ao NMS. São os equipamentos de rede: routers, switches, computadores, entre outros. Neste trabalho serão os routers e switches da rede.

A figura seguinte representa a típica arquitectura de gestão rede [CISCO2008]:

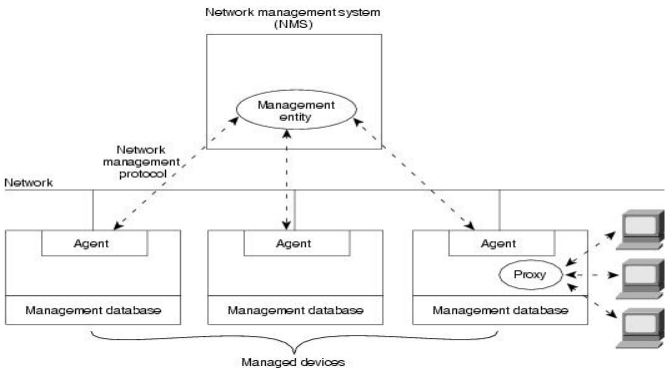


Figura 59 - Relações NMS, agentes e MD

O conjunto de todos os objectos SNMP é conhecido por Management Information Base (MIB). Esta é estruturada em árvore, com informações no topo e ramificando-se nos detalhes. A definição de objectos é independente do protocolo de comunicação, permitindo criar novos conjuntos de variáveis MIB, definidos como *standards*, para novos dispositivos ou novos protocolos. Por isso, foram criados conjuntos de variáveis MIB que correspondem a protocolos como UDP, IP, ARP, assim como variáveis MIB para *Ethernet* ou FDDI. Cada fabricante pode implementar a sua própria MIB para definir objectos que devem ser geridos. Existem alguns padrões sugeridos, mas a ideia é que cada um incorpore objectos numa MIB conforme a necessidade.

O transporte de informação entre NMS e o agente é não orientado à conexão – UDP. A escolha deste protocolo de transporte foi baseada no facto de que no UDP, apesar de não se ter garantia na comunicação, há menos *overhead* e assim não se sobrecarrega tanto a rede com o sistema de gestão. Os portos utilizados são o 161 para envio de pedido e recebimento de resposta e o porto 162 para recebimento de *traps*. Especificamente neste trabalho o mecanismo de transferência de informação utilizado é o *polling* (pedido e resposta). O gestor faz pedidos ao agente periodicamente.

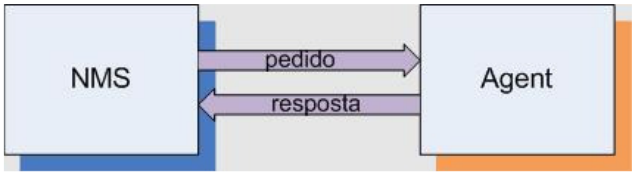


Figura 60 - Transporte de informação entre NMS e Agente

A segurança não é o forte nas versões SNMPv1 e SNMPv2 (a primeira utilizada neste trabalho), pois baseiam-se em *community strings*, que são simples *passwords*, *strings* em formato texto aberto, que permitem que qualquer ferramenta de gerência que conheça esta *string* obtenha acesso aos dados deste dispositivo. O agente pode ser configurado com três tipos de *community: read-only, read-write* e *trap*. A *community read-only* permite apenas leitura de dados no agente, a *read-write* permite leitura e modificação de dados e valores no agente, e *trap* permite envio de informações do agente para o NMS de forma assíncrona.

5.2.1 Linguagem de definição dos dados

É necessário assegurar que a informação de gestão seja representada de forma consistente. A estrutura adoptada é a SMI (Structure of Management Information) que possui dois standards: SMIv1 e SMIv2. A informação de gestão é representada por escalares e tabelas (escalares 2D) pois o protocolo SNMP só permite trocar listas de escalares.

5.2.1.1 SMIv1

Structure of Management Information 1, RFC 1155, define como os objectos geridos são denominados e qual tipo de dados estão associados. Resumindo em três atributos temos:

- Name – nome ou OID (Object Identifier) define um objecto como único. Uma hierarquia em árvore, a ISO Object Identifier tree, foi a solução encontrada para nomear todos os objectos (protocolos, dados, etc). Os nós da árvore representam uma subdivisão por organização ou função. Os valores das variáveis da MIB são guardados nas folhas da árvore. Assim, a cada variável corresponde um único caminho a partir da raiz. Os filhos de um nó são numerados sequencialmente da esquerda para a direita a começar pelo 1, separados por pontos, de modo a que cada nó da árvore tenha um nome único, que consiste no aglutinar de todos os números que compõem o caminho desde a raiz até o nó em causa.

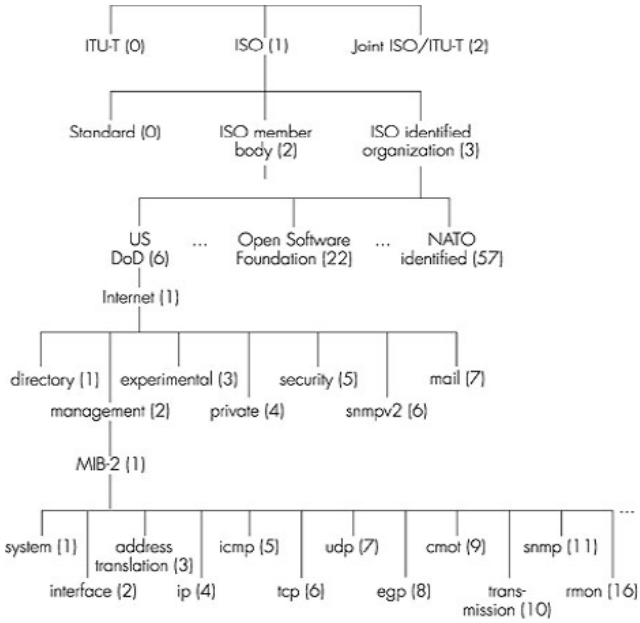


Figura 61 - OSI Object Identifier Tree

Por exemplo, percorrendo a árvore iso(1) identified-organization(3) dod(6) internet(1) management(2) mib-2(1) system(1) temos a representação deste OID na forma 1.3.6.1.2.1.1.

- Type – O tipo de dados é definido usando notações do ASN.1 (Abstract Syntax Notation One) [ASN2008]. É uma notação formal para descrever informação transmitida através de um protocolo de comunicação, independentemente da linguagem de implementação, da representação física da informação, da aplicação, e independente da complexidade. Os tipos de dados podem ser Integer, Unsigned,

Octet String, Object Identifier, Sequence, IPaddress, Couter, Guage, TimeTicks, Opaque.

- Encoding - Há algumas formas de codificação de dados estandardizados tais como a codificação BER (Basic Encoding Rules). BER define a forma como os objectos são codificados e descodificados para a transmissão e pode permitir cumprir requisitos a nível de largura de banda

5.2.1.2 SMIv2

Structure of Management Information 2, RFC 2578, amplia a árvore de objectos adicionando o braço snmpv2 à sub-árvore internet. Adiciona novos tipos de dados tais como Integer32, Counter32, Gauge32, Unsigned32, Counter64, Bits. Modifica a definição de um objecto, adicionando novos campos, proporciona maior controlo sobre a forma como um objecto é acedido e permite aumentar uma tabela adicionando mais colunas oferecendo portanto melhores descrições.

5.2.2 Mensagens SNMP

As mensagens entre NMS e agentes são:

- Get: é uma mensagem enviada do gestor para o agente, fazendo um pedido do valor da variável, através da identificação do objecto.
- GetNext: o gestor pede informação do objecto seguinte ao que é enviado na mensagem.
- Set: o gestor envia uma mensagem que transporta o valor que deseja colocar no objecto alvo.
- Response: mensagem cujo conteúdo é o valor da variável pedida anteriormente.
- Trap: o agente informa o gestor de informações não solicitadas, um evento excepcional.

No âmbito deste trabalho, foi utilizada a **Walk** que representa sucessivos **GetNext**. O funcionamento do get-next basea-se em percorrer a árvore SMI até encontrar o OID desejado, enviando um get-next a cada get-response. Após localizar o OID desejado ele continua enviando requisições **GetNext** até receber um erro, neste ponto ele finaliza a busca.

As mensagens SNMP são constituídas por três campos: a versão SNMP, a community string e o PDU. O PDU é composto por vários campos. Nas figuras seguintes apresenta-se o formato do PDU nas mensagens Get, Set, Response e Trap:

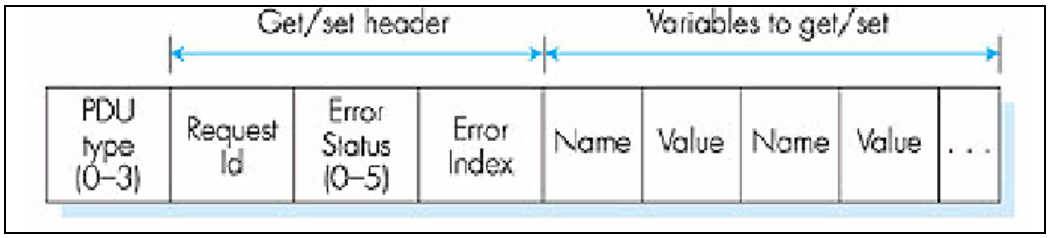


Figura 62 - PDU mensagens Get, Set e Response

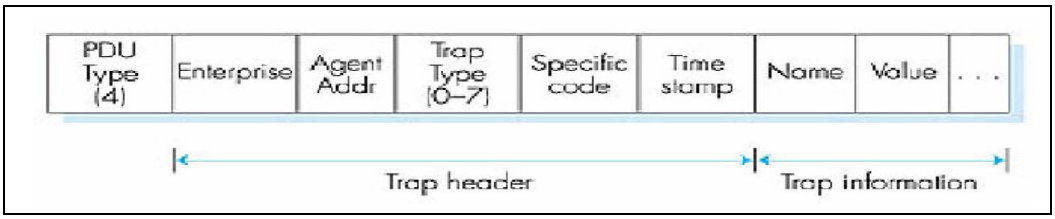


Figura 63 - PDU mensagem Trap

5.3 Switch Processing Scripts

Através do script **script_switch.sh** fez-se o processamento dos dados do switch, que anteriormente foram recolhidos através do script já referido **script_snmp.sh**. Através deste processamento os dados são limpos e armazenados na base de dados. Por exemplo, os dados da figura 7, depois de limpos, ficam como na figura seguinte:

2 Cisco IOS Software, C3750ME Software (C3750ME-I5-M), Version 12.2(25)SEG1, RELEASE SOFTWARE (fc1) Copyright (c) 1986-2006 by Cisco Systems, Inc. Mon 07-Aug-06 20:26 by myl SNMPv2-SMI::enterprises.9.1.574 0:10:19 pmoreira@av.it.pt Switch4 IT Network Lab. 1 6 2008-05-13 10:42:01

Figura 64 - Saída equipamento

Como se pode observar os dados estão separados por tabulações para poderem ser carregados para a base de dados.

No processamento reorganizaram-se os dados em sete grupos. Os dados de cada grupo depois de processados são armazenados em ficheiros cuja nomenclatura apresenta o parâmetro **$1** (parâmetro de entrada do **script_switch.sh**), que é o nome do equipamento.

O processamento de cada um destes grupos é descrito nos sub-capítulos seguintes e em todos estes grupos constarão os campos **id_equip, data_dia** e **hora**. O **id_equip** é o identificador do equipamento. É obtido através do comando da figura seguinte:

```
id_equip=`mysql -u $user -p$pass_mysql --skip-column-names -e "SELECT id FROM
list_of_equipment WHERE name_equip = '$1'" equipment `
```

Figura 65 - Comando para obter o identificador do equipamento

O **script_snmp.sh**, quando executado, recolhe os dados dos equipamentos por SNMP e escreve a data e hora actual na base de dados **equipment**, tabela **list-of-equipment**. Assim, quando o **script_switch.sh** é executado, lê da mesma base de dados o valor dos campos **data_dia** e **hora**,

que correspondem a data e hora que houve a última recolha dos dados pois, estes dados são os do equipamento na hora da recolha. Os comandos utilizados são apresentados de seguida:

```
data_dia=`mysql -u $user -p$pass_mysql --skip-column-names -e "SELECT last_date_get
FROM list_of_equipment WHERE name_equip = '$1'" equipment `
hora=`mysql -u $user -p$pass_mysql --skip-column-names -e "SELECT last_hour_get
FROM list_of_equipment WHERE name_equip = '$1'" equipment `
```

Figura 66 - Comando para obter data_dia e hora

Os dados de cada grupo são posteriormente carregados para tabelas, **description**, **interfaces**, **ip_interfaces**, **datagram_ip**, **icmp**, **tcp_udp** e **ip_porto_udp**, na base de dados do equipamento $1. As tabelas serão descritas no capítulo Base de dados. Os grupos são descritos de seguida.

5.3.1 System

Descreve características do equipamento, como por exemplo: o nome, sistema operativo, localização, entre outros. Dados limpos são escritos no ficheiro **saida_"$1"_system.txt**.

Para a limpeza dos dados foi utilizada a função da figura seguinte:

```
function answer1()
{
cat $path | grep $1 | cut -d"=" -f 2 > $path_auxiliar
}
```

Figura 67 - Função answer1

Chama-se a função através do comando **answer1 $1.** O **$1** é o parâmetro de entrada da função **answer1** e esta retorna todo o conteúdo que está depois do primeiro **"="** nas linhas que apresentam a palavra **$1**.

Na tabela seguinte, em cada coluna, descreve-se a coluna correspondente no ficheiro de saída; palavras-chave utilizadas (**$1**); a notação utilizada no script para estes dados; qual o significado destes; e os comandos utilizados adicionais à função **answer1**.

Coluna	**$1**	**Notação script**	**Significado**	**Comandos adicionais (comando=** cat path_auxiliar)
1	-	.	Receberá o identificador da linha na base de dados	-
2	-	id_equip	é o identificador do serviço numa determinada máquina	obtido com o comando da figura 65
3	Descr	system	descrição do equipamento	`**comando** | cut -d":" -f 2`
4	Support	tech_support	endereço suporte técnico	` **comando** | cut -d" " -f 3 `
5	Copyright	copyright	licensa	`**comando** `

6	Compiled	compiled	data da compilação	`**comando** | awk '{print $2" "$3" "$4" "$5" "$6}``
7	ObjectID	id	identificação do vendedor	`**comando** | cut -d" " -f 3`
8	UpTimeInstance	time	tempo desde de que foi reinicializado pela última vez	`**comando** | cut -d" " -f 4 | cut -d"." -f 1 `
9	Contact	contact	contacto do administrador do equipamento	`**comando** | cut -d":" -f 2 | cut -d" " -f 2`
10	Name	name	nome administrativo do equipamento	`**comando** | cut -d":" -f 2 `
11	Location	location	localização física	`**comando** | cut -d":" -f 2 `
12	Services	services	valor que indica os serviços que este equipamento oferece	`**comando** | cut -d":" -f 2 | cut -d" " -f 2`
13	-	data_dia	data da última recolha dos dados do equipamento	Lê da base de dados com o comando da figura 66
14	-	hora	hora da última recolha dos dados do equipamento	Lê da base de dados com o comando da figura 66

Tabela 32 - Dados ficheiro saída_$1_system

5.3.2 Interface

As informações são relativas a cada interface do equipamento, tais como MAC, pacotes enviados e recebidos. Dados são escritos no ficheiro **saida_"$1"_interface.txt**.

Para a limpeza dos dados foram utilizadas as funções seguintes:

```
function answer()
{
cat $path1 | grep $1 | head -n 1 | cut -d"=" -f 2 | cut -d":" -f 2 | cut -d" " -f 2 > $path_auxiliar
}
```

Figura 68 - Função answer

```
function answer3()
{
cat $path3 | grep $1 | head -n 1 | cut -d"=" -f 2 | cut -d":" -f 2 | cut -d" " -f 2 > $path_auxiliar
}
```

Figura 69 - Função answer3

Chamam-se as funções através dos comandos **answer $1 e anwer3 $1.** As funções como se pode observar são iguais, apenas o path é alterado, pois um path corresponde aos dados retirados da RFC1213-MIB e outro aos dados da IF-MIB. O **$1** é o parâmetro de entrada das funções e retornam todo o conteúdo que está depois do primeiro **"="**, depois do **":"** e depois do carácter espaço, nas linhas que apresentam a palavra **$1**.

Um detalhe importante neste processamento é que para o processamento deste grupo foi preciso ter sempre em conta o índex da interface, conforme coluna 1 da tabela seguinte. Logo, o parâmetro de entrada das funções **$1** será sempre **palavra-chave.$value_of_interface.** Um trecho do RFC1213-MIB é apresentado na coluna 2 da tabela seguinte e pode-se observar aqui a relação entre os MAC e os indexes das interfaces e para o resto dos dados comporta-se da mesma forma.

OID .1.3.6.1.2.1.2.2.1.1	**OID .1.3.6.1.2.1.2.2.1.6**
IF-MIB::ifIndex.1 = INTEGER: 1	IF-MIB::ifPhysAddress.1 = STRING: 0:1a:a2:3f:d5:40
IF-MIB::ifIndex.10001 = INTEGER: 10001	IF-MIB::ifPhysAddress.10001 = STRING: 0:1a:a2:3f:d5:3
IF-MIB::ifIndex.10002 = INTEGER: 10002	IF-MIB::ifPhysAddress.10002 = STRING: 0:1a:a2:3f:d5:4
IF-MIB::ifIndex.10003 = INTEGER: 10003	IF-MIB::ifPhysAddress.10003 = STRING: 0:1a:a2:3f:d5:5
IF-MIB::ifIndex.10004 = INTEGER: 10004	IF-MIB::ifPhysAddress.10004 = STRING: 0:1a:a2:3f:d5:6
IF-MIB::ifIndex.10005 = INTEGER: 10005	IF-MIB::ifPhysAddress.10005 = STRING: 0:1a:a2:3f:d5:7
IF-MIB::ifIndex.10006 = INTEGER: 10006	IF-MIB::ifPhysAddress.10006 = STRING: 0:1a:a2:3f:d5:8
IF-MIB::ifIndex.10007 = INTEGER: 10007	IF-MIB::ifPhysAddress.10007 = STRING: 0:1a:a2:3f:d5:9
IF-MIB::ifIndex.10008 = INTEGER: 10008	IF-MIB::ifPhysAddress.10008 = STRING: 0:1a:a2:3f:d5:a
IF-MIB::ifIndex.10009 = INTEGER: 10009	IF-MIB::ifPhysAddress.10009 = STRING: 0:1a:a2:3f:d5:b
IF-MIB::ifIndex.10010 = INTEGER: 10010	IF-MIB::ifPhysAddress.10010 = STRING: 0:1a:a2:3f:d5:c
IF-MIB::ifIndex.10011 = INTEGER: 10011	IF-MIB::ifPhysAddress.10011 = STRING: 0:1a:a2:3f:d5:d
IF-MIB::ifIndex.10012 = INTEGER: 10012	IF-MIB::ifPhysAddress.10012 = STRING: 0:1a:a2:3f:d5:e
IF-MIB::ifIndex.10013 = INTEGER: 10013	IF-MIB::ifPhysAddress.10013 = STRING: 0:1a:a2:3f:d5:f
IF-MIB::ifIndex.10014 = INTEGER: 10014	IF-MIB::ifPhysAddress.10014 = STRING: 0:1a:a2:3f:d5:10
IF-MIB::ifIndex.10015 = INTEGER: 10015	IF-MIB::ifPhysAddress.10015 = STRING: 0:1a:a2:3f:d5:11
IF-MIB::ifIndex.10016 = INTEGER: 10016	IF-MIB::ifPhysAddress.10016 = STRING: 0:1a:a2:3f:d5:12
IF-MIB::ifIndex.10017 = INTEGER: 10017	IF-MIB::ifPhysAddress.10017 = STRING: 0:1a:a2:3f:d5:13
IF-MIB::ifIndex.10018 = INTEGER: 10018	IF-MIB::ifPhysAddress.10018 = STRING: 0:1a:a2:3f:d5:14
IF-MIB::ifIndex.10019 = INTEGER: 10019	IF-MIB::ifPhysAddress.10019 = STRING: 0:1a:a2:3f:d5:15
IF-MIB::ifIndex.10020 = INTEGER: 10020	IF-MIB::ifPhysAddress.10020 = STRING: 0:1a:a2:3f:d5:16
IF-MIB::ifIndex.10021 = INTEGER:	IF-MIB::ifPhysAddress.10021 = STRING:

10021 IF-MIB::ifIndex.10022 = INTEGER: 10022 IF-MIB::ifIndex.10023 = INTEGER: 10023 IF-MIB::ifIndex.10024 = INTEGER: 10024 IF-MIB::ifIndex.10101 = INTEGER: 10101 IF-MIB::ifIndex.10102 = INTEGER: 10102 IF-MIB::ifIndex.10301 = INTEGER: 10301 IF-MIB::ifIndex.10302 = INTEGER: 10302	0:1a:a2:3f:d5:17 IF-MIB::ifPhysAddress.10022 = STRING: 0:1a:a2:3f:d5:18 IF-MIB::ifPhysAddress.10023 = STRING: 0:1a:a2:3f:d5:19 IF-MIB::ifPhysAddress.10024 = STRING: 0:1a:a2:3f:d5:1a IF-MIB::ifPhysAddress.10101 = STRING: 0:1a:a2:3f:d5:1 IF-MIB::ifPhysAddress.10102 = STRING: 0:1a:a2:3f:d5:2 IF-MIB::ifPhysAddress.10301 = STRING: 0:1a:a2:3f:d5:1b IF-MIB::ifPhysAddress.10302 = STRING: 0:1a:a2:3f:d5:1c

Tabela 33 - Informações em função da interface

Assim, para se conhecer os números das interfaces **($value_of_interface),** primeiramente fez-se um array com estes números utilizando o comando **value=(`cat $path1 | grep ifIndex | cut -d'' '' -f 1 | cut -d''.'' -f 2`)**. Depois, percorre-se todo o array para aceder aos dados.

As funções utilizam também o comando **head -n 1**, uma vez que, para o número da interface apenas interessa o primeiro caso do **grep**. Pois, por exemplo, para obter os dados da interface **1**, com um **grep palavra-chave.1** também se iria obter todos os dados das interfaces cujos números começassem pelo número 1.

Na tabela seguinte, em cada coluna, descreve-se: a coluna correspondente no ficheiro de saída; palavras-chave utilizadas; a notação utilizada no script para estes dados; qual o significado destes.

Coluna	**Palavra-chave**	**Notação script**	**Significado**
1	-	.	Receberá o identificador da linha na base de dados
2	-	id_equip	é o identificador do serviço numa determinada máquina
3	**value**	value_of_interface	número da interface
4	ifDescr	tecnology_of_interface	tipo de "tecnologia" da interface
5	ifType	type_of_interface	tipo de interface
6	ifMtu	mtu_of_interface	tamanho máximo do datagrama que pode ser recebido ou enviado pela interface
7	ifSpeed	speed_of_interface	estimativa da actual largura de banda em bits/segundo
8	ifPhysAddress	mac_of_interface	mac da interface

9	ifAdminStatus	admin_status_of_interface	interface
10	ifOperStatus	status_of_interface	actual estado da interface
11	ifLastChange	lastchange_of_interface	o valor do sysUpTime quando a interface esta operacional
12	ifInOctets	in_octets	número de octetos recebido na interface
13	ifInUcastPkts	in_unicast_packets	número de pacotes unicast enviados para a camada superior do protocolo
14	ifInNUcastPkts	in_no_unicast_packets	número de pacotes não unicast enviados para a camada superior do protocolo
15	ifInDiscards	in_discarded_packets	número de pacotes de entrada que foram escolhidos para serem descartados
16	ifInErrors	in_error_packets	número de pacotes de entrada que continham erros
17	ifInUnknownProtos	unknown_packets	número de pacotes recebidos pela interface que foram descartados porque não eram conhecidos ou por ser de um protocolo não suportado
18	ifInUnknownProtos	out_octets	número de octetos enviados pela interface
19	ifOutUcastPkts	out_unicast_packets	número de pacotes que a camada superior do protocolo requisitou para a transmissão para um endereço unicast, inclui os descartados e não enviados
20	ifOutNUcastPkts	out_no_unicast_packets	número de pacotes que a camada superior do protocolo requisitou para a transmissão para um endereço não unicast, inclui os descartados e não enviados
21	ifOutDiscards	out_discarded_packets	número de pacotes de saída que foram escolhidos para serem descartados
22	ifOutErrors	out_error_packets	número de pacotes de

			saída que não puderam ser transmitidos devido a erros
23	ifOutQLen	length_of_queue	tamanho da fila de saída dos pacotes(em pacotes)
24	ifInMulticastPkts	in_multicast_packets	numero de pacotes enviados para uma camada de rede superior com endereço multicast
25	ifInBroadcastPkts	in_broadcast_packets	número de pacotes enviados para uma camada de rede superior com endereço broadcast
26	ifOutMulticastPkts	out_multicast_packets	número de pacotes que a camada de rede superior pediu para ser transmitido com endereço multicast, incluindo os descartados e não enviados
27	ifOutBroadcastPkts	out_broadcast_packets	número de pacotes que a camada de rede superior pediu para ser transmitido com endereço broadcast, incluindo os descartados e não enviados
28	ifLinkUpDownTrapEnable	link_up_down	indica se traps podem ser gerados para esta interface
29	ifHighSpeed	high_speed	estimativa da largura de banda da interface em 1,000,000 bits/segundo
30	ifPromiscuousMode	promiscuous_mode	se for falso(ter valor 2) é porque a interface apenas aceita pacotes e frames que são endereçadas a esta estação. tem o valor de 1 se a estação aceita todos os pacotes e frames
31	ifConnectorPresent	connector	é verdadeiro(1) se a interface tem um conector físico e falso(2) se não tiver.
32	ifCounterDiscontinuityTime	counter_discontinuity_time	o valor mais recente do sysUpTime quando qualquer uma ou mais interfaces sofreram descontinuidade
33	-	data_dia	data da última recolha

			dos dados do equipamento
34	-	hora	hora da última recolha dos dados do equipamento

Tabela 34 - Dados ficheiro saída_$1_interface.txt

O campo **id_equip** é obtido através do comando da figura 65. Os campos **data_dia** e **hora** são lidos da base de dados através do comando da figura 66. Os campos descritos da coluna 4 a 23 do script (linha 4 a 23 da tabela anterior) são obtidos através da função **answer** e da coluna 24 a 32 do script (linha 24 a 32 da tabela anterior) com a função **answer3**.

5.3.3 Ip

As informações aqui retiradas são sempre relativas ao endereço IP da interface do equipamento ligado ao equipamento em questão. Correspondência entre indexs, endereços IP pertencentes ao equipamento ou ligados a ele, MAC, máscara e outros. Os dados são escritos no ficheiro **saida_"$1"_ip.txt**.

Para a limpeza dos dados foram utilizadas diferentes funções e dados de diferentes partes das MIBs, agrupando-se tudo que apresentava o IP da interface do equipamento que está ligado a este. As funções são as das figuras seguintes:

```
function answer4()
{
rm $path_auxiliar2
cat $path2 | grep $1 | cut -d"=" -f 2 | cut -d":" -f 2 >> $path_auxiliar
}
```

Figura 70 - Função answer4

```
function answer6()
{
rm $path_auxiliar
cat $path1 | grep $1 | cut -d"=" -f 2 | cut -d":" -f 2 >> $path_auxiliar
}
```

Figura 71 - Função answer6

Chama-se a função através do comando **answer4/6 $1.** Neste caso o **$1** é o parâmetro de entrada da função e esta retorna todo o conteúdo que está depois do primeiro **"="**e do ":" nas linhas que apresentam a palavra **$1**.

Na tabela seguinte, em cada coluna, descreve-se a coluna correspondente no ficheiro de saída; palavras-chave utilizadas (**$1**); a notação utilizada no script para estes dados; qual o significado destes.

Coluna	$1	Notação script	Significado
1	-	.	Receberá o identificador da linha na base de dados
2	-	id_equip	é o identificador do serviço numa determinada máquina
3	atIfIndex	indexs	Número da interface do equipamento
4	-	ip	IP da interface do equipamento ligado a este
5	atPhysAddress	mac	MAC da interface do equipamento ligado a este
6	atNetAddress	network_address	endereço ip em hexadecimal
7	ipAdEntNetMask	mask	máscara de sub-rede associada
8	ipAdEntBcastAddr	bit_bcast	valor do bit menos significativo no endereço IP broadcast. Este valor é 1 quando é usado broadcast
9	ipAdEntReasmMaxSize	max_size	o tamanho do maior datagrama IP que este equipamento pode aceitar
10	ipNetToMediaType	media_type	tipo de mapa
13	-	data_dia	data da última recolha dos dados do equipamento
14	-	hora	hora da última recolha dos dados do equipamento

Tabela 35 - Dados ficheiro saída_$1_ip.txt

Na figura seguinte temos um trecho do ficheiro de onde retirou-se o campo ip (IP da interface do equipamento ligado a este):

```
RFC1213-MIB::atIfIndex.1.1.10.0.0.4 = INTEGER: 1
RFC1213-MIB::atIfIndex.1.1.10.0.0.5 = INTEGER: 1
RFC1213-MIB::atIfIndex.1.1.10.0.0.6 = INTEGER: 1
RFC1213-MIB::atIfIndex.1.1.192.168.11.2 = INTEGER: 1
```

Figura 72 - Obter IP

Retira-se assim o ip com o comando **cat $path1 | grep atIfIndex | cut -d"=" -f 1** e concatena-se a informação entre os **“.”**. O campo **id_equip** é obtido através do comando da figura 65. Os campos **data_dia** e **hora** são lidos da base de dados através do comando da figura 66.

5.3.4 Datagramas

Descreve as estatísticas dos datagramas IP. Dados são escritos no ficheiro **saida_”$1”_datagramas.txt**.

Para a limpeza dos dados foi utilizada a função da figura seguinte:

```
function answer2()
{
cat $path2 | grep $1 | cut -d"=" -f 2 | cut -d":" -f 2 > $path_auxiliar
}
```

Figura 73 - Função answer2

Chama-se a função através do comando **answer2 $1.** O **$1** é o parâmetro de entrada da função **answer2** e esta retorna todo o conteúdo que está depois do primeiro **"="** e do ":" nas linhas que apresentam a palavra **$1**.

Na tabela seguinte, em cada coluna, descreve-se: a coluna correspondente no ficheiro de saída; palavras-chave utilizadas (**$1**); a notação utilizada no script para estes dados; qual o significado destes.

Coluna	**$1**	**Notação script**	**Significado**
1	-	.	Receberá o identificador da linha na base de dados
2	-	id_equip	é o identificador do serviço numa determinada máquina
3	ipForwarding	forwarding	indica se está ou não a haver transmissão
4	ipDefaultTTL	ttl	valor default do campo time-to-live do cabeçalho ip originário deste aparelho
5	ipInReceives	in_receives	número total de datagramas recebidos pelas interfaces, incluindo os com erro
6	ipInHdrErrors	in_error	número de datagramas recebidos descartados devido a variados erros
7	ipInAddrErrors	in_addr_error	número de datagramas recebidos descartados porque o endereço ip não era válido
8	ipForwDatagrams	forw_datagrams	número de datagramas recebidos em que o ip deste equipamento(neste caso switch4) não é o ip destino final
9	ipInUnknownProtos	in_unknown_prot	número de datagramas recebidos com sucesso mas descartados devido a um desconhecido ou não suportado protocolo
10	ipInDiscards	in_discard	número de datagramas recebidos que não apresentavam nenhum problema mas foram descartados(ex por falta de espaço no buffer)
11	ipInDelivers	in_delivers	número de datagramas recebidos com sucesso enviados por um IP protocolo(inclui ICMP)
12	ipOutRequests	out_request	número total de datagramas IP requisitados para transmissão. não inclui os datagramas contados em ipForwdatagrams
13	ipOutDiscards	out_discards	número de datagramas IP requisitados que não apresentavam problema mas que foram descartados
14	ipOutNoRoutes	out_no_routes	número de datagramas ip descartados pois não foi encontrado roteamento para transmiti-los até ao destino
15	ipReasmTimeout	reasm_timeout	número máximo de segundos que recebeu-se fragmentos enquanto estes estavam à espera de serem

			reassemblados neste equipamento
16	ipReasmReqds	reasm_req	número de fragmentos IP que necessitam de ser reassemblados neste equipamento
17	ipReasmOKs	reasm_ok	número de datagramas IP reassemblados com sucesso
18	ipReasmFails	reasm_fail	número de falhas detectadas por um algoritmo IP
19	ipFragOKs	frag_ok	número de datagramas IP que foram fragmentados com sucesso neste equipamento
20	ipFragFails	frag_fail	número de datagramas IP que foram descartados pois precisavam de ser reassemblados neste equipamento mas não poderiam ser pois a flag Don't Fragment estava activa
21	ipFragCreates	frag_creates	número de fragmentos de datagramas IP que foram gerados como resultado da fragmentação neste equipamento
22	-	data_dia	data da última recolha dos dados do equipamento
23	-	hora	hora da última recolha dos dados do equipamento

Tabela 36 - Dados ficheiro saída_$1_datagramas.txt

O campo **id_equip** é obtido através do comando da figura 65. Os campos **data_dia** e **hora** são lidos da base de dados através do comando da figura 66.

5.3.5 Icmp

Descreve estatísticas dos pacotes ICMP. Dados são escritos no ficheiro **saida_"$1"_icmp.txt**.

Para a limpeza dos dados foi utilizada a função da figura seguinte:

```
function answer7()
{
cat $path4 | grep $1 | cut -d"=" -f 2 | cut -d":" -f 2 > $path_auxiliar
}
```

Figura 74 - Função answer7

Chama-se a função através do comando **answer7 $1.** Semelhante a função **answer7**, porém o path dos dados é outro. O **$1** é o parâmetro de entrada da função esta retorna o conteúdo que está depois do primeiro **"="**, depois do **":"** nas linhas que apresentam a palavra **$1**.

Na tabela seguinte, em cada coluna, descreve-se: a coluna correspondente no ficheiro de saída; palavras-chave utilizadas (**$1**); a notação utilizada no script para estes dados; qual o significado destes.

Coluna	$1	Notação script	Significado
1	-	.	Receberá o identificador da linha na base de dados
2	-	id_equip	é o identificador do serviço numa determinada máquina
3	icmpInMsgs	in_icmp	número total de ICMP messages que o equipamento recebeu
4	icmpInErrors	in_error	número de icmp messages que recebeu com erros
5	icmpInDestUnreachs	in_dest_unr	número de icmp destination unreachable recebidos
6	icmpInRedirects	in_redirect	número de icmp redirect mensagens recebidos
7	icmpInEchos	in_echo	número de icmp echo (request) recebidos
8	icmpInEchoReps	in_echo_rep	número de icmp echo reply recebidos
9	icmpOutMsgs	out_icmp	número total de icmp mensagens que este equipamento enviou
10	icmpOutErrors	out_error	número de icmp messages que não enviou pois haviam erros
11	icmpOutDestUnreachs	out_dest_unr	número de icmp destination unreachable enviados
12	icmpOutRedirects	out_redirect	número de icmp redirect mensagens enviadas
13	icmpOutEchos	out_echo	número de icmp echo (request) enviados
14	icmpOutEchoReps	out_echo_rep	número de icmp echo reply recebidos
15	-	data_dia	data da última recolha dos dados do equipamento
16	-	hora	hora da última recolha dos dados do equipamento

Tabela 37 - Dados ficheiro saída_$1_icmp.txt

O campo **id_equip** é obtido através do comando da figura 65. Os campos **data_dia** e **hora** são lidos da base de dados através do comando da figura 66.

5.3.6 Estatisticas tcp e udp

Descreve as estatísticas dos pacotes e datagramas TCP e UDP. Dados são escritos no ficheiro **saida_"$1"_est_tcp_udp.txt**.

Para a limpeza dos dados foi utilizada também a função **answer7** descrita na figura 74.

Na tabela seguinte, em cada coluna, descreve-se: a coluna correspondente no ficheiro de saída; palavras-chave utilizadas (**$1**); a notação utilizada no script para estes dados; qual o significado destes.

Coluna	$1	Notação script	Significado
1	-	.	Receberá o identificador da linha na base de dados
2	-	id_equip	é o identificador do serviço numa determinada máquina
3	tcpMaxConn	tcp_conn_max	máximo número de conexões tcp neste equipamento. se o valor for -1 é porque o numero máximo de conexões é dinâmico
4	tcpInSegs	tcp_in	número total de segmentos recebidos, incluindo os com erros
5	tcpOutSegs	tcp_out	número de segmentos enviados, mas excluído os que contém apenas octetos retransmitidos
6	tcpRetransSegs	tcp_retr	número de segmentos retransmitidos
7	tcpInErrs	tcp_in_err	número de segmentos recebidos com erro
8	udpInDatagrams	udp_in	número total de UDP datagramas recebidos
9	udpNoPorts	udp_no_port	número de UDP datagramas recebidos que não tem nenhuma aplicação na porta destino
10	udpInErrors	udp_in_err	número de UDP datagramas recebidos que não foram enviados por outras razões sem ser a falta de aplicação na porta destino
11	udpOutDatagrams	udp_out_dat	número total de datagramas UDP enviados deste equipamento
12	-	data_dia	data da última recolha dos dados do equipamento
13	-	hora	hora da última recolha dos dados do equipamento

Tabela 38 - Dados ficheiro saída_$1_est_tcp_udp.txt

5.3.7 Portos udp

Descreve os portos UDP ocupados no equipamento. Os dados são escritos no ficheiro **saida_"$1"_udp.txt**.

Para a limpeza dos dados foi utilizada a função da figura seguinte:

```
function answer8()
{
rm $path_auxiliar
cat $path4 | grep $1 | cut -d"=" -f 2 | cut -d":" -f 2 | cut -d" " -f 2 >> $path_auxiliar
}
```

Figura 75 - Função answer8

Chama-se a função através do comando **answer8 $1.** O **$1** é o parâmetro de entrada da função e esta retorna todo o conteúdo que está depois do primeiro **"=",** depois do **":"** e depois do carácter espaço, nas linhas que apresentam a palavra **$1**. Semelhante a função **answer7**, porém o

encaminhamento para o ficheiro auxiliar adiciona linhas a este, e não substitui como na outra função.

Na tabela seguinte, em cada coluna, descreve-se a coluna correspondente no ficheiro de saída; palavras-chave utilizadas (**$1**); a notação utilizada no script para estes dados; qual o significado destes.

Coluna	**$1**	**Notação script**	**Significado**
1	-	.	Receberá o identificador da linha na base de dados
2	-	id_equip	é o identificador do serviço numa determinada máquina
3	udpLocalAddress	udp_address	endereço ip que ouve UDP
4	udpLocalPort	udp_port	número da porta deste ouvinte UDP
5	-	data_dia	data da última recolha dos dados do equipamento
6	-	hora	hora da última recolha dos dados do equipamento

Tabela 39 - Dados ficheiro saída_$1_udp.txt

O campo **id_equip** é obtido através do comando da figura 65. Os campos **data_dia** e **hora** são lidos da base de dados através do comando da figura 66.

5.4 Router Processing Scripts

Através do script **script_router.sh** há o processamento dos dados do router, recolhidos através do script já referido **script_snmp.sh**.

Os dados foram reorganizados em oito grupos: os mesmos sete dos switch e mais um descrito no sub-capítulo seguinte.

5.4.1 Rede

Apresenta informações sempre em relação a rede e métricas utilizadas no roteamento. Dados são escritos no ficheiro **saida_"$1"_rede.txt**. Na nomenclatura o **$1** significa o nome do equipamento. Os dados, em colunas separados por tabulações no ficheiro, deste grupo são posteriormente carregados para a tabela **network** na base de dados do equipamento $1. A tabela será descrita no capítulo Base de dados.

Para a limpeza dos dados foi utilizada a função answer 4 da figura 70 e esta é chamada da mesma forma referida anteriormente.

Na tabela seguinte, em cada coluna, descreve-se: a coluna correspondente no ficheiro de saída; palavras-chave utilizadas (**$1**); a notação utilizada no script para estes dados; qual o significado destes.

Coluna	$1	Notação script	Significado
1	-	.	Receberá o identificador da linha na base de dados
2	-	id_equip	é o identificador do serviço numa determinada máquina
3	ipRouteDest	rede	IP da rede
4	ipRouteIfIndex	route_index	identificador da interface
5	ipRouteMetric1	metric1	métrica primária para o roteamento.Se esta métrica não for usada o seu valor é -1
6	ipRouteMetric2	metric2	métrica alternativa para o roteamento.Se esta métrica não for usada o seu valor é -1
7	ipRouteMetric3	metric3	métrica alternativa para o roteamento.Se esta métrica não for usada o seu valor é -1
8	ipRouteMetric4	metric4	métrica alternativa para o roteamento.Se esta métrica não for usada o seu valor é -1
9	ipRouteMetric5	metric5	métrica alternativa para o roteamento. Se esta métrica não for usada o seu valor é -1
10	ipRouteNextHop	next_hop	endereço ip do próximo salto
11	ipRouteType	route_type	tipo de roteamento. Pode ser directo(3) ou indirecto(4), ou inválido(2).
12	ipRouteProto	proto	tipo de mecanismo de *routing* em que este equipamento aprendeu. Pode ser: other, local, netmgmt, icmp, egp, ggp, hello, rip, is-is,es-is, ciscoIgrp, bbnSpfIgp, ospf, bgp
13	ipRouteAge	route_age	número de segundos desde que este equipamento sofreu um updated
14	ipRouteMask	route_mask	máscara da rede
15	-	data_dia	data da última recolha dos dados do equipamento
16	-	hora	hora da última recolha dos dados do equipamento

Tabela 40 - Dados ficheiro saída_$1_rede.txt

O identificador do equipamento é obtido também através do comando da figura 65, assim como, os campos **data_dia** e **hora** com os comandos da figura 66.

5.5 Síntese

Neste capítulo apresentou-se o protocolo SNMP e a sua importância na monitorização de redes. Pretendeu-se também, fazer a análise e limpeza de toda a informação recolhida dos equipamentos de rede através deste protocolo, de forma a clarificar e acrescentar outras informações que possam vir a ser fundamentais na detecção de anomalias na rede. Para isto foram utilizados os comandos de shell script descritos nos sub-capítulos 5.3 e 5.4.

6 Base de dados

6.1 Introdução

Uma base de dados é uma colecção de dados estruturados, organizados e armazenados de uma forma persistente. Para adicionar, aceder e processar dados armazenados na base de dados é necessário um sistema de gestão SGBD (Sistema de Gestão de Base de Dados) como o servidor MySQL.

6.2 Servidor MySQL

É um servidor robusto de base de dados, utiliza SQL (Structured Query Language), bastante rápido, multi-tarefa e multi-usuário, RDBMS (Relational Database Management System). Foi usado neste trabalho sob os termos da GNU General Public License.

O MySQL tornou-se disponível publicamente em 1996, mas o inicio de seu desenvolvimento data de 1979. Este, foi desenvolvido originalmente para lidar com bases de dados muito grandes de forma muito mais rápida do que as soluções existentes. Oferece um rico conjunto de funções e a sua conectividade, velocidade e segurança são características bastante positivas.

A instalação foi feita através do comando **sudo apt-get install mysql-server**. A versão MySQL utilizada foi a 5.0.38.

O Webmin serviu apenas como auxiliar de consulta na visualização das tabelas criadas, nomes das colunas e tipos durante a elaboração do trabalho.

Nos sub-capítulos seguintes descrevem-se os tipos de campos, restrições e comandos utilizados [MySQL2008].

6.2.1 Tipos de campos

O MySQL suporta tipos de campos que podem ser agrupados em três categorias: tipos numéricos, de data e hora e tipos caracteres. De seguida, são descritos os tipos de campos utilizados neste trabalho:

Notas: UNSIGNED, isto é, números sem sinal, foram usados sempre nos tipos numéricos deste trabalho pois, não há nenhuma variável negativa. M, determina o número máximo de dígitos que o número ou string contém.

TINYINT(M) UNSIGNED: inteiro muito pequeno que sem sinal tem a sua gama de 0 a 255.

SMALLINT(M) UNSIGNED: inteiro pequeno com gama, sem sinal, de 0 a 65535.

INT(M) UNSIGNED: inteiro cuja gama sem sinal varia entre 0 a 4294967295.

BIGINT(M) UNSIGNED: inteiro grande. A gama sem sinal é de 0 a 18446744073709551615.

DATE: data com o formato yyyy-mm-dd. A gama suportada é de 1000-01-01 a 9999-12-31.

TIME: hora. Formato seguido é HH:MM:SS.

CHAR(M): string de tamanho fixo que quando armazenada é preenchida a direita com espaços até o tamanho especificado. Os espaços extras são removidos quando o valor é recuperado. M, tamanho máximo de dígitos (display), pode variar entre 1 e 255. Mas, a partir da versão 4.1.0, se M é superior a 255 o tipo é convertido para TEXT.

VARCHAR(M): string de tamanho variável. M aqui também pode variar entre 1 e 255. São armazenados usando apenas o numero de caracteres estritamente necessário, isto é, os espaços extras são removidos quando os valores são armazenados e mais um byte para armazenar o tamanho.

Tanto para o CHAR quanto VARCHAR, se o valor exceder o tamanho máximo da coluna, o valor é truncado para este tamanho.

6.2.2 Restrições

São as regras a que os valores de uma ou mais colunas devem obedecer. Logo, todas as linhas que se pretendem inserir numa tabela terão que obedecer ao conjunto de restrições existentes para as colunas dessa tabela. As restrições utilizadas neste trabalho são:

NOT NULL: impede a introdução de valores nulos na coluna especificada.

PRIMARY KEY: o conteúdo da coluna não pode ser nulo e não pode admitir repetições. Utiliza-se o atributo AUTO INCREMENT para criar sempre uma identificação única.

REFERENCES: esta restrição permite fazer a validação das chaves estrangeiras. Isto é, não se podem introduzir nos campos referenciados como chave estrangeira, valores que não existam na tabela onde os campos são chave primária. Se houver necessidade de referenciar uma chave estrangeira formada por dois ou mais campos, terá que se utilizar a cláusula FOREIGN KEY.

6.2.3 Comandos

A seguir estão descritos os comandos utilizados neste trabalho. As respectivas sintaxes encontram-se no Anexo VI.

- SELECT: utilizado para retornar registos seleccionados de uma ou mais tabelas.

Um exemplo utilizado num shell script, **script_get_log.sh,** para obter o ip é o da seguinte figura:

```
ip_serv=`mysql -u $user -p$pass_mysql --skip-column-names -e "SELECT ip_server FROM list_of_servers WHERE id = '$1'" servers `
```

Figura 76 - Exemplo comando SELECT

Retorna-se assim da tabela list_of_servers o valor da coluna ip_server e com a restrição feita pela cláusula WHERE especifica-se a linha desejada e sucessivamente o valor para o ip. Neste caso, o comando encontra-se embutido num comando shell, logo é necessário dizer também qual a base de dados (no exemplo é a **servers**) e especificar as permissões de acesso à base de dados: o username e password, variáveis **$user** e **$pass_mysql** respectivamente.

O comando SELECT foi utilizado não só nos shell scripts como no código PHP. Exemplo retirado do **client_result.php**:

```
$query = "SELECT * FROM $a WHERE ip = '$ipc' AND date > = '$day'";
```

Figura 77 - Exemplo SELECT no código PHP

A cláusula DISTINCT também foi utilizada para criar tabelas temporárias que são usadas na interface *Web*, script PHP. São estas: tabela com todos os serviços, tabela com todos os sistemas operativos e outra com todos os endereços IP dos servidores disponíveis. Os dados apresentam-se ordenados e sem repetição. O exemplo a seguir descreve uma das situações descritas:

```
$sql1 = "SELECT DISTINCT serv from list_of_servers";
```

Figura 78 - Exemplo cláusula DISTINCT

Na interface será apresentada uma toolbox com a lista de todos os serviços disponíveis.

- UPDATE: actualiza uma coluna em registos de tabelas existentes. Sintaxe é descrita no Anexo VI.

A cláusula SET indica quais as colunas a modificar e o seu novo valor e a cláusula WHERE especifica quais linhas devem ser actualizadas.

Segue-se um exemplo utilizado no script **script_get_log.sh** para actualizar a data que houve a recolha dos ficheiros de log no servidor:

```
mysql -u $user -p$pass_mysql --skip-column-names -e "UPDATE list_of_servers SET last_date_get = '$date_get' WHERE id = '$1'" servers
```

Figura 79 - Exemplo comando UPDATE

Actualiza-se assim na tabela list_of_servers a coluna last_date_get através da cláusula SET, e com o WHERE especifica-se que apenas a linha com o id especificado deve ser actualizada.

O comando UPDATE também foi utilizado nos scripts PHP. Exemplo retirado do **actualizar_servidores_equipamentos.php**:

```
$sql2= "UPDATE list_of_servers SET username = '$value_user' WHERE id = '$id_server' ";
```

Figura 80 - Exemplo comando UPDATE

- DELETE: apaga linhas de uma tabela ou toda a tabela. Sintaxe é descrita no Anexo VI.

Na figura seguinte observa-se um exemplo utilizado no script PHP **actualizar_servidores_equipamentos.php**.

```
$sql4= "DELETE FROM list_of_servers WHERE id = '$id_server' ";
```

Figura 81 - Exemplo comando DELETE

Neste exemplo, a linha que satisfaz a cláusula WHERE é apagada da tabela list_of_servers.

- CREATE DATABASE: cria uma base de dados com o nome especificado.

Como exemplo temos no script .sql **script_base_dados_apache.sql**:

```
CREATE DATABASE IF NOT EXISTS servers;
```

Figura 82 - Exemplo comando CREATE

Neste exemplo, temos que a base de dados com o nome servers será criada, se ainda não existir, no servidor MySQL.

- USE: comando diz ao MySQL para usar a base de dados indicada como padrão para as consultas subsequentes. Esta base de dados continua como actual até o final da sessão ou até outra instrução USE ser executada.

Como exemplo temos no mesmo script:

```
USE servers;
```

Figura 83 - Exemplo comando USE

Neste exemplo o comando irá dizer ao MySQL para usar a base de dados servers.

- CREATE TABLE: cria uma tabela, se ainda não existir, com o nome especificado na base de dados actual.

Como exemplo temos no script .sql **script_base_dados_apache.sql**:

```
CREATE TABLE IF NOT EXISTS apache ( id_line BIGINT UNSIGNED NOT NULL AUTO_INCREMENT, id_server TINYINT(3) UNSIGNED NOT NULL REFERENCES list_of_servers(id), date DATE, hour TIME, ip VARCHAR(15), done VARCHAR(50), status_server TINYINT UNSIGNED, browser VARCHAR(20), so VARCHAR(20), bytes INT UNSIGNED, PRIMARY KEY (id_line));
```

Figura 84 - Exemplo comando CREATE TABLE

Através deste comando é criada uma tabela com o nome apache e com as colunas e tipos de dados especificados.

LOAD DATA INFILE: lê linhas de um ficheiro de texto e carrega os dados presentes para a tabela especificada. Para o correcto carregamento dos dados, o ficheiro de texto deve ter os dados em colunas separadas por tabulação.

Quando LOCAL é especificado, o ficheiro é lido pelo programa cliente na máquina cliente e enviada ao servidor. Neste trabalho, LOCAL não é especificado pois o ficheiro esta localizado na máquina servidora e é lido directamente pelo servidor (LOCAL está disponível no MySQL versão 3.22.6 ou posterior).

Como exemplo, temos no mesmo script .sql do exemplo anterior:

```
LOAD DATA INFILE "$path/saida_apache.txt" INTO TABLE apache;
```

Figura 85 - Exemplo comando LOAD DATA INFILE

Através deste comando carregam-se os dados do ficheiro de texto que contém a saída dos dados do serviço apache para a tabela apache.

6.3 Modelação da Base de Dados

Neste trabalho segue-se o modelo relacional, em que a estrutura de dados utilizada, como o próprio nome diz, é a relação. Neste modelo a única forma de relacionar os dados que existem numa tabela com dados de outra tabela é através de atributos comuns.

Com o auxílio do software Dbdesigner4 e de uma das suas ferramentas, Reverse Engineering, representou-se a estrutura criada da base de dados relativamente aos utilizadores da interface *Web* (figura 86), e serviços e equipamentos existentes numa rede (figura 87 e 88 respectivamente).

6.3.1 Dbdesigner4

É uma ferramenta de modelação de base de dados livre e sob a licença GNU GPL License. Integra design de base de dados, modelagem, criação. Suporta MySQL, SQLite, Oracle, MSSQL e ODBC. Permite definir a relação entre as tabelas, recurso a engenharia reversa e sincronização do modelo com a base de dados. Disponível para sistemas Linux e Windows.

6.3.2 Modelação lista de utilizadores

Foi criada, através do script SQL **script_users.sql**, uma base de dados, a **list_of_users**, e nela uma tabela, **users**, que contém as informações dos administradores que podem ter acesso a esta interface.

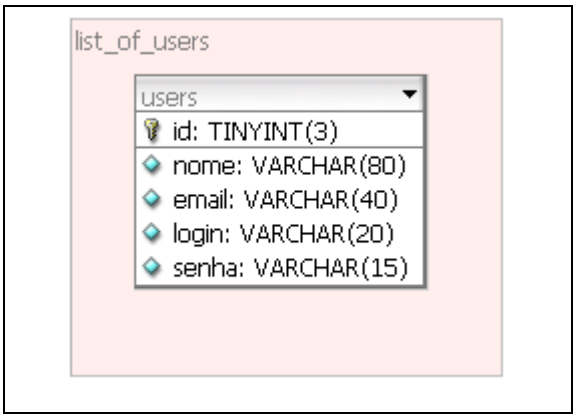


Figura 86 - Modelação base de dados utilizadores

O identificador de cada usuário é um PRIMARY KEY e é acrescentado através do atributo AUTO INCREMENT. Na tabela são armazenados o nome, e-mail, login e senha dos utilizadores (administradores).

6.3.3 Modelação serviços

Para definir a bases de dados para todos os serviços criados utilizamos as seguintes tabelas dentro de uma base de dados também criada chamada **servers**:

- Criou-se uma tabela para cada serviço descrito no capítulo Monitorização de serviços através dos scripts anteriormente referenciados no sub-capítulo 3.4.2. O conteúdo destas tabelas correspondem aos campos descritos nas tabelas dos sub-capítulos ***nome_do_serviço*** **Data Processing Script** do capítulo 4.
- Criou-se uma tabela **list_of_servers** que contém os principais dados dos serviços monitorizados da rede tais como: o nome do servidor, do serviço que presta, do SO da máquina servidor, *username* e *password* para aceder ao servidor (usado na recolha dos ficheiros de log, SCP - descrito no sub-capítulo 3.3.1), endereços IP e MAC do servidor, caminho em que encontra-se seu ficheiro de log na maquina servidora e caminho destino para este ficheiro de log na maquina do administrador, a data da última vez que lhe foram alterados os campos anteriores, o intervalo de tempo entre recolha e processamento dos logs, o ultimo dia e data da recolha dos logs, o ultimo dia e data do processamento dos logs.

Na figura seguinte observam-se as tabelas referidas:

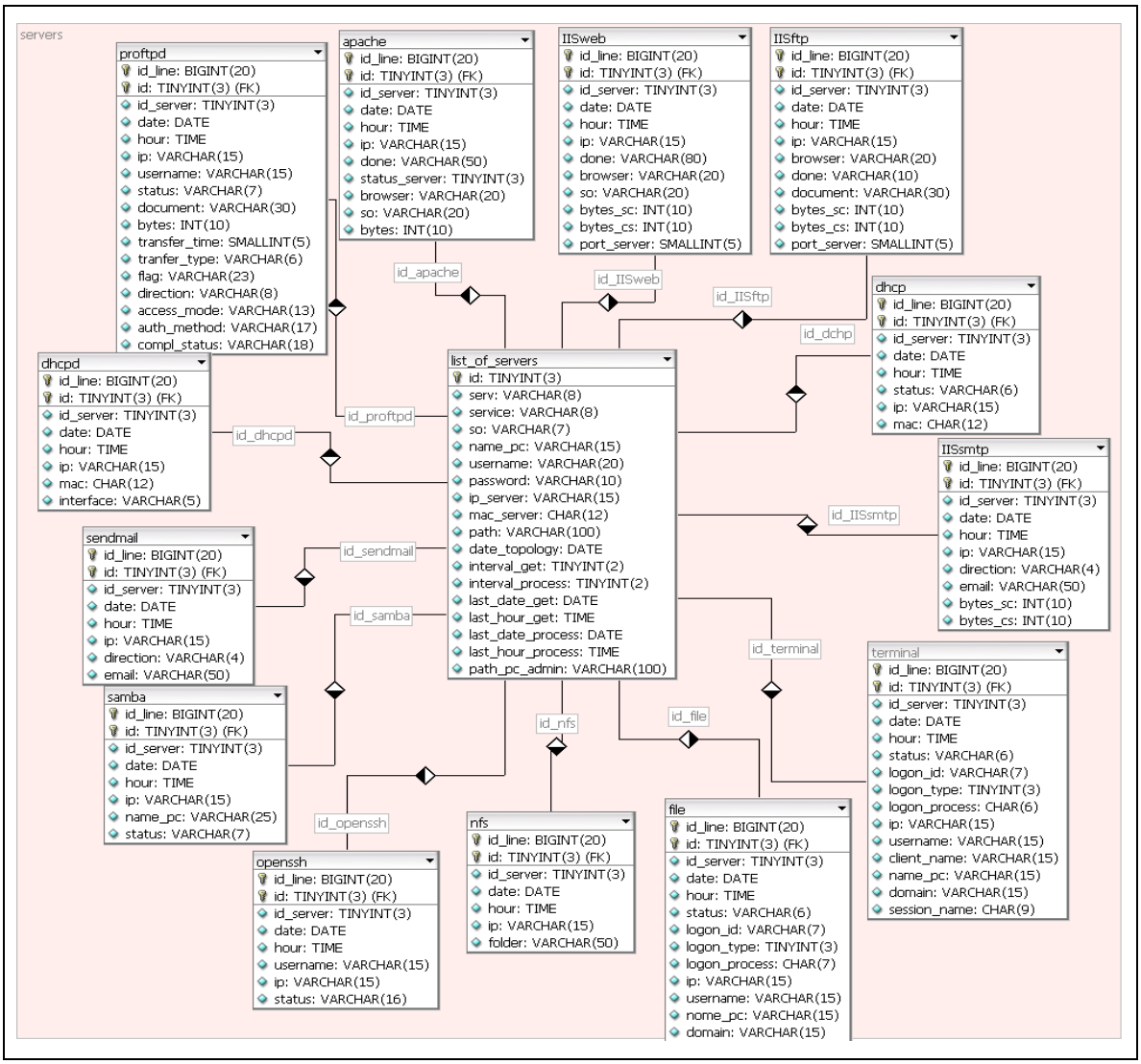


Figura 87 - Modelação base de dados servidores

No esquema da figura anterior é possível observar que são definidas relações 1:N (um-para-muitos) entre as entidades **id**, na tabela **list_of_servers**, e **id_server**, nas tabelas dos serviços. Desta forma, a entidade **id_server** recebe como atributo a chave primária da entidade **id**. Nota-se que o **id_server** não é chave-primária. Trata-se de um atributo comum e este é denominado chave-estrangeira.

Declararam-se o número de servidores, **id** na tabela **list_of_servers**, como TINYINT (3) supondo-se que não há mais do que 255 servidores numa rede.

6.3.4 Modelação equipamentos de rede

Criou-se a base de dados **equipment** e nela uma tabela denominada **list_of_equipment** que contém os principais dados dos equipamentos que o administrador pretende monitorizar na sua rede. Estes dados são: o nome do equipamento, o tipo (router ou switch), a *community string* configurada (usada na recolha dos dados, SNMP - descrito no sub-capítulo 3.3.2), endereço IP, a

data da última vez que lhe foram alterados os campos anteriores, o intervalo de tempo entre recolha e processamento dos dados, o ultimo dia e data da recolha dos dados, o último dia e data do processamento dos dados.

Para definir a base de dados dos equipamentos na rede (routers ou switches) cria-se uma base de dados no servidor MySQL para cada equipamento. O nome de cada uma é o nome do equipamento. Posteriormente, dentro da base de dados, são criadas as tabelas definidas na figura seguinte – figura 88. Cada tabela contém os dados dos ficheiros explicados no capítulo Monitorização dos equipamentos de rede: **saida_"$1"_system.txt**, **saida_"$1"_interface.txt**, **saida_"$1"_ip.txt**, **saida_"$1"_datagramas.txt**, **saida_"$1"_icmp.txt**, **saida_"$1"_est_tcp_udp.txt**, **saida_"$1"_udp.txt** e **saida_"$1"_rede.txt** (este último apenas para os routers).

Este processo é todo ele feito através de shell scripts e sql scripts, que verificam na tabela **list_of_equipments** quais os equipamentos na rede que o administrador quer monitorizar. Posteriormente, verifica se há alguma base de dados com o nome do equipamento, e se não houver, cria a base de dados e tabelas respectivas. Na figura seguinte observam-se as tabelas referidas:

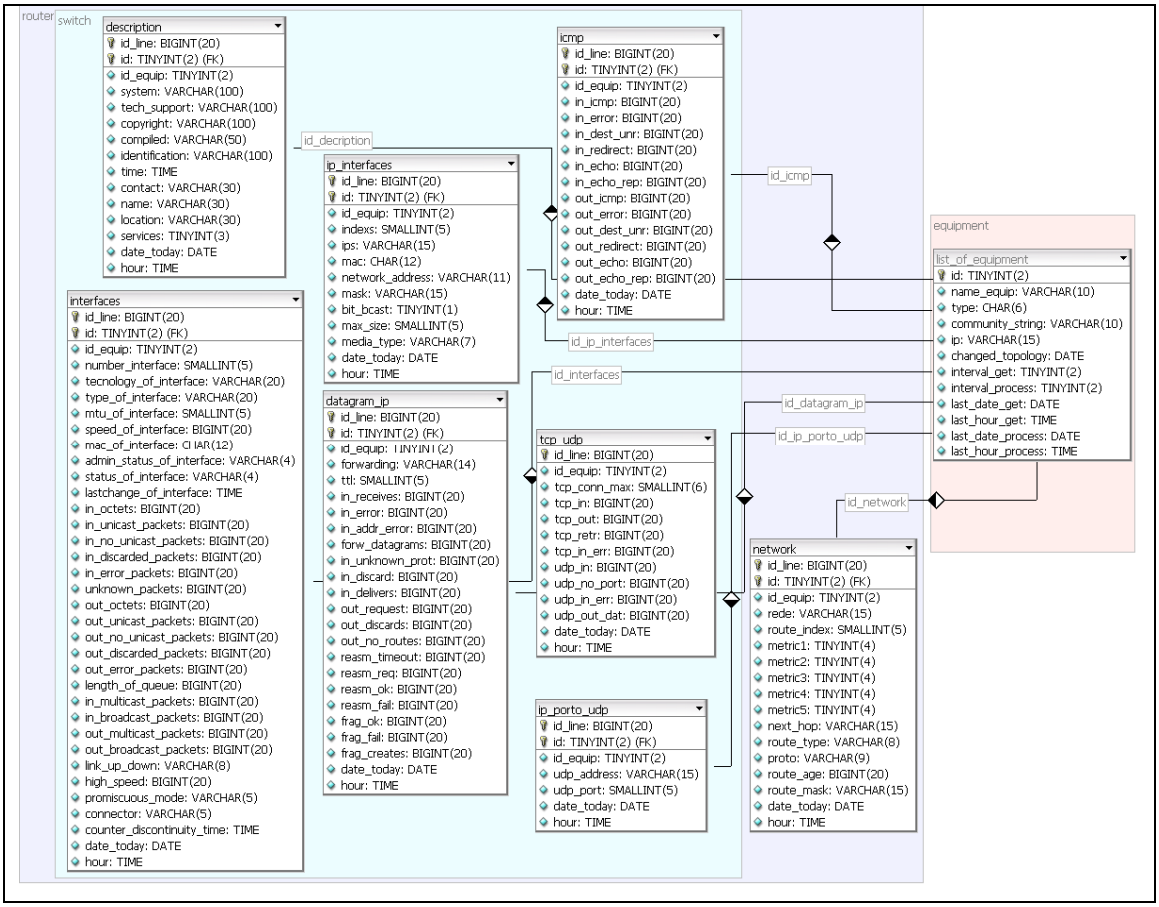


Figura 88 - Modelação base de dados equipamentos

São definidas relações 1:N (um-para-muitos) entre as entidades **id**, na tabela **list_of_equipment**, e **id_equip**, nas tabelas dos serviços. Desta forma, a entidade **id_equip** recebe como atributo a chave primária da entidade **id**.

Declarou-se o número de equipamentos como TINYINT (3) pois, supõe-se que não há mais do que 255 equipamentos numa rede.

6.4 Síntese

Neste capítulo apresentou-se o servidor de base de dados MySQL, os seus campos, restrições e comandos utilizados neste trabalho. Apresentou-se também, a modelação das bases de dados utilizadas no armazenamento dos dados recolhidos de servidores e equipamentos de rede e dos utilizadores da interface *Web*.

7 Interface Web

7.1 Introdução

Desenvolveu-se uma interface *Web* para uso do administrador da rede. Nesta interface o administrador tem um total controlo sob as acções do sistema descrito na figura 1 e realizado neste trabalho. Neste capítulo será apresentado de que forma o administrador agenda os serviços e equipamentos de rede a serem monitorizados e como os dados destes são observados através de diferentes filtros.

Esta interface foi desenvolvida em linguagem PHP e HTML.

7.2 PHP

PHP significa PHP Hypertext Preprocessor, surgiu em 1994, foi criado por Rasmus Lerdof e originalmente chamava-se Personal Home Page. É uma *server-side scripting* linguagem de programação orientada a objectos, interpretada, livre, bastante modularizada e utilizada para desenvolvimento *Web*. Pode ser embebida em HTML ou usada em *standalone binary*. O código PHP é interpretado pelo servidor *Web* e gera os outputs para os clientes.

As melhores vantagens do PHP são:

- Lidar com diferentes bases de dados de uma forma fácil, dentre elas: Oracle, PostgreSQL, SQLite, Firebird, MySQL.
- Óptimo desempenho.
- Portabilidade, disponível em diferentes sistemas operativos.
- Ter um conjunto de funções com uma estrutura flexível de programação.
- Fácil aprendizagem pois é baseada em outras linguagens de programação como C e Perl.
- Suportar inúmeros protocolos: IMAP, SNMP, POP3, HTTP, NNTP.

A versão utilizada neste trabalho foi a PHP5.2.5 e os passos da instalação encontram-se no Anexo VII.

7.3 Frames utilizadas

Na figura seguinte apresenta-se um diagrama que descreve a estrutura da página.

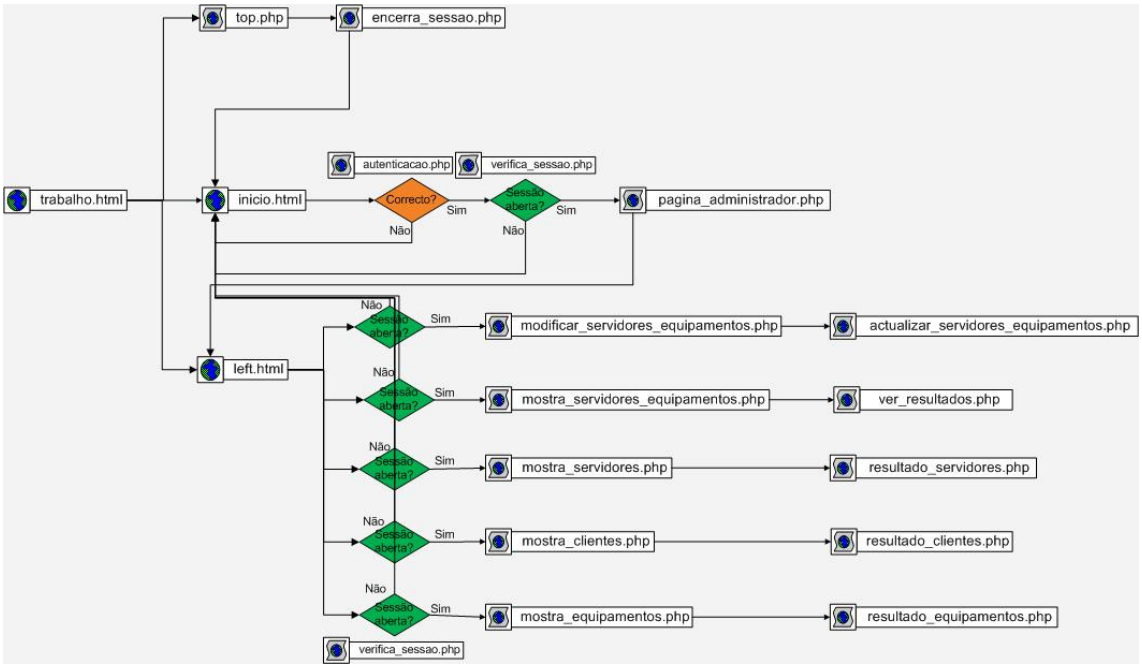


Figura 89 - Fluxograma interface Web

Foram sempre utilizadas três *frames*. Porém, ao longo da documentação da página, apenas a *frame* central vai sendo alterada. As frames centrais distinguem-se em três cores: cor esverdeada representa o login do utilizador, cor branca representa área da página em que o administrador pode modificar ou inserir novos serviços e equipamentos do processo de monitorização, cor azul são as frames de visualização das informações recolhidas. As outras duas *frames*, **top.php** e **left.html**, mantêm-se e a última contém os *links* do conteúdo da página.

Assim que se tem acesso à interface *Web*, o administrador deve preencher obrigatoriamente o seu login e *password*. Na figura seguinte apresenta-se esta *frame*, **inicio.html**:

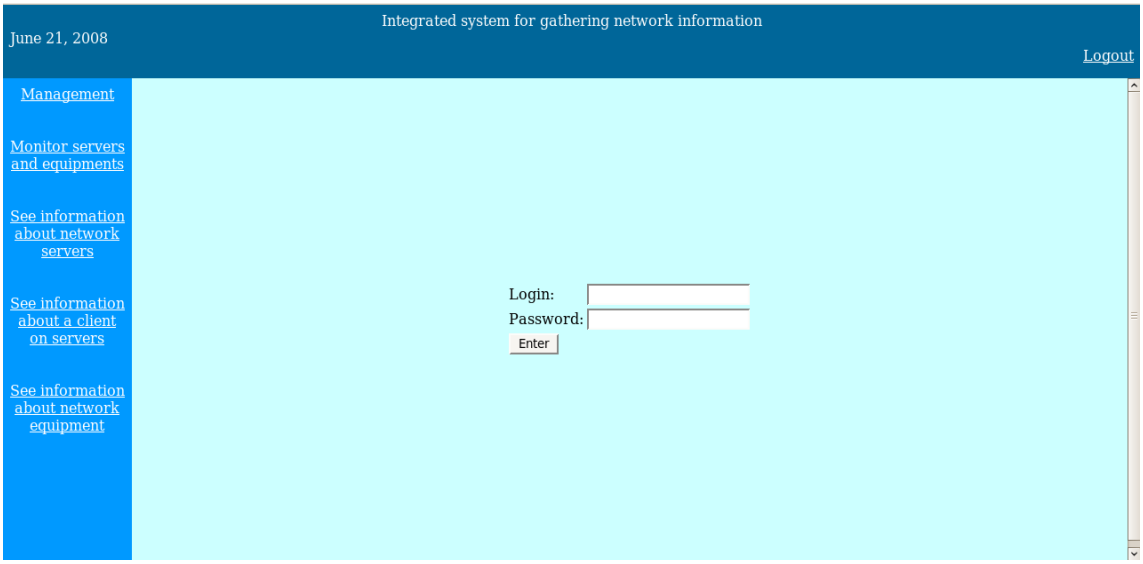


Figura 90 - inicio.html

Os dados introduzidos são enviados por POST para o script de autenticação **autenticacao.php**. O último script recebe os dados e procura na base de dados **list_of_users**, tabela **users**, se há alguma linha com os dados correspondentes através do comando na figura seguinte:

```
$sql = "SELECT id, nome FROM users WHERE login = '$login' AND senha = '$senha'";
```

Figura 91 - Consulta de autenticação

Não é permitido o acesso a qualquer dos *links* se antes estes dados não passarem por este processo de validação. Se não forem válidos, emite-se um alerta e deve-se voltar a introduzir os dados do utilizador.

Após o preenchimento e validação é aberta uma **sessão**. Sempre que o administrador navega pela página é previamente verificado se há uma sessão aberta ou não através do **verifica_sessao.php**. Assim, o administrador tem total controlo sobre à página, até que resolva terminar a sua sessão em qualquer momento através do *link* **Logout** na frame superior **top.php**. Na frame da figura 92 são dadas indicações ao administrador do que este pode fazer através desta interface. Este conteúdo apresenta-se na figura seguinte:

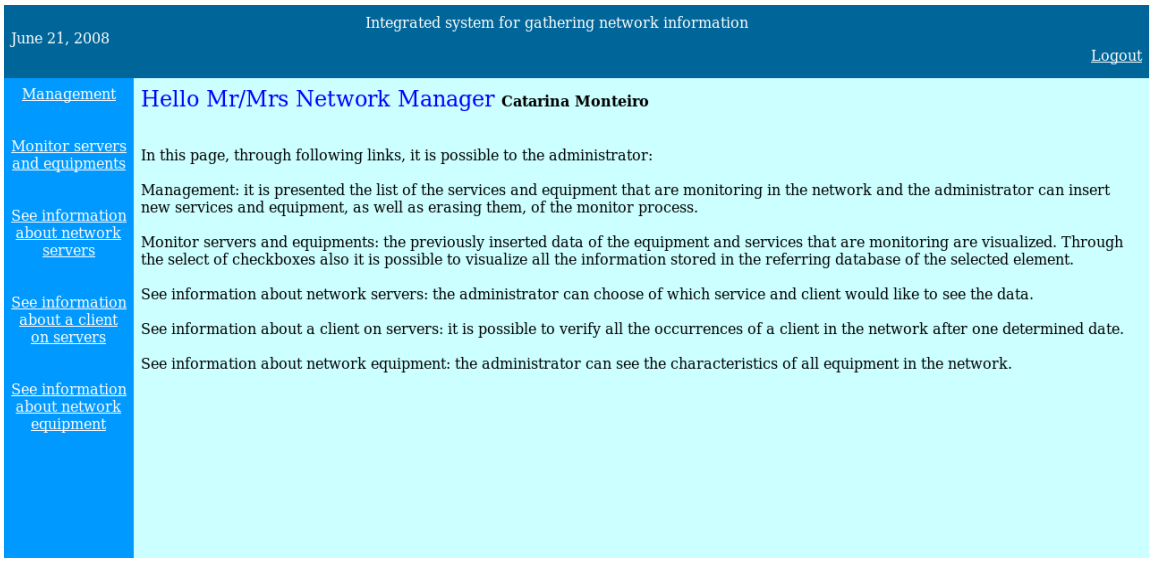


Figura 92 - pagina_administrador.html

Há a disposição do administrador através da interface *Web* cinco *links* (na frame esquerda **left.html**): **Management**, **Monitor servers and equipments**, **See information about network servers**, **See information about a client on servers**, **See information about network equipment**. A descrição de cada um é feita nos parágrafos seguintes.

Através do *link* **Management** é apresentada a lista dos serviços e equipamentos que estão a ser monitorizados na rede (tabelas **list_of_servers** e **list_of_equipments** presentes na base de dados **servers** e **equipments** respectivamente). O administrador pode inserir serviços e equipamentos, assim como apagá-los do processo de monitorização.

Na tabela correspondente aos serviços, é ainda possível modificar o IP e MAC do servidor, o nome da máquina servidora, o *username* e *password* de acesso ao servidor, os caminhos para os ficheiros de log na máquina servidora e o caminho onde estes serão armazenados na máquina do administrador. Se for feita alguma mudança destes parâmetros será actualizada a data descrita na coluna **date that changed characteristics of the server.** Também através desta tabela são configurados os intervalos de tempo que o administrador deseja que os ficheiros de log sejam recolhidos do servidor para a sua máquina, **get logs at intervals of** e o intervalo entre processamentos sucessivos, **process logs at intervals of**. São apresentados ao administrador alguns dados que não podem ser alterados: o sistema operativo da máquina servidora e a data e hora da última recolha de logs e respectivos processamentos.

Na tabela correspondente aos equipamentos é possível modificar o endereço IP do equipamento na rede, sua community string (utilizada para leitura dos dados através do protocolo SNMP), os intervalos de tempo que o administrador deseja ler os dados, **get data at intervals of**, e o intervalo entre os respectivos processamentos sucessivos dos dados, **process data at intervals of**. A quando destas alterações é actualizada a data descrita na coluna **date that changed characteristics of the equipment**. Os dados que não podem ser alterados são: o tipo de equipamento de rede (router/switch) e a data e hora da última recolha dos dados e respectivos processamentos.

Na figura seguinte é apresentada a frame descrita acima:

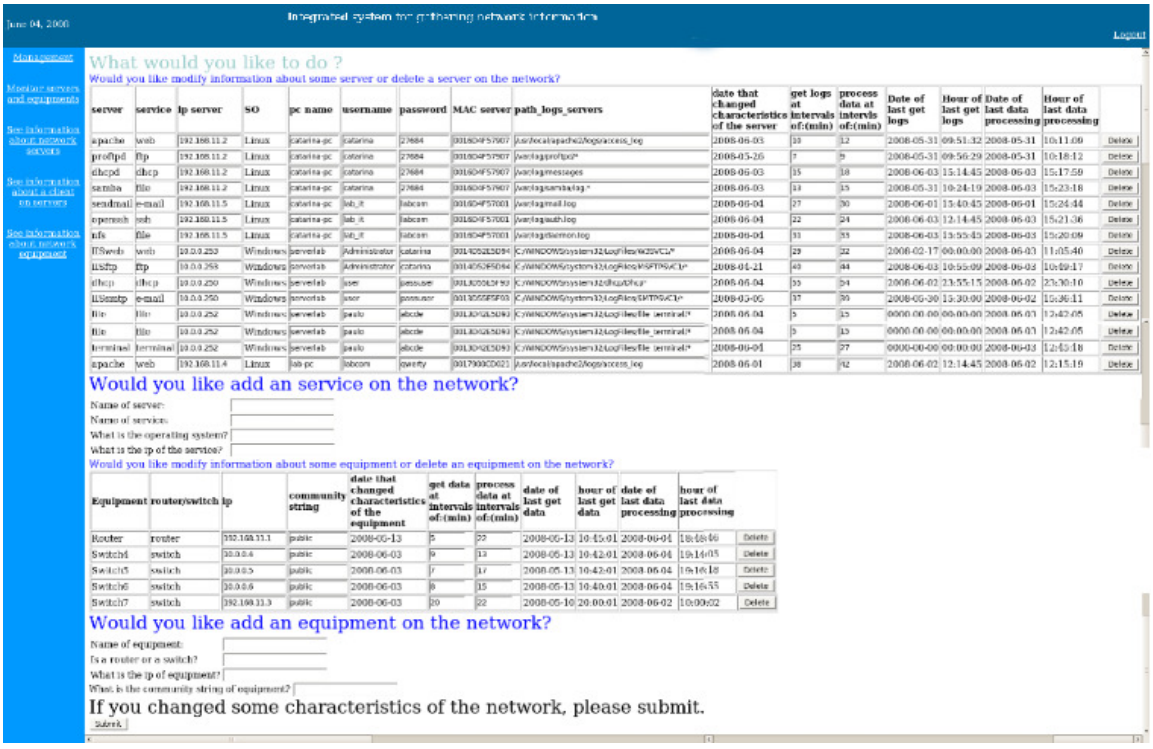


Figura 93 - modificar_servidores_equipamentos.php

Após a alteração dos dados presentes ou inserção de novos, é feito um POST para a frame seguinte (figura 94) e há a actualização na base de dados destes novos dados, através de comandos UPDATE. Teve-se em atenção a verificação se os dados anteriores eram ou não iguais aos novos. Através de comandos SELECT lê-se os dados antigos e compara-se com os novos dados, se forem iguais não há a escrita na base de dados.

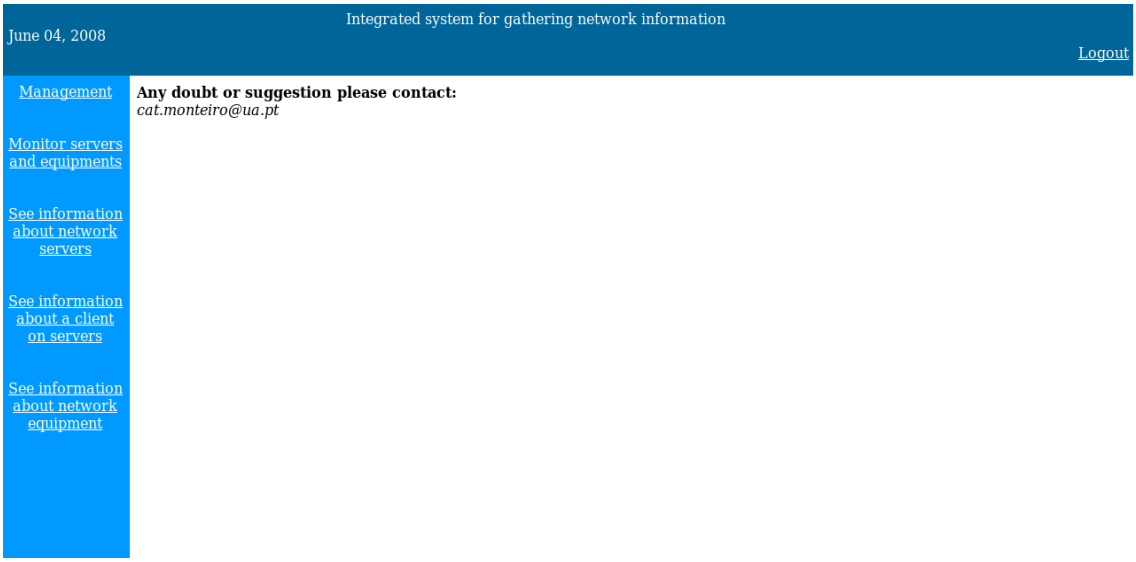


Figura 94 - actualizar_servidores_equipamentos.php

Através do **Monitor servers and equipments** o administrador visualiza os dados anteriormente inseridos e que estão na base de dados, tabelas **list_of_servers** e **list_of_equipments**. Através da selecção das checkboxes é possível visualizar todas as informações armazenadas na base de dados referentes ao elemento seleccionado. Observa-se esta frame na figura seguinte:

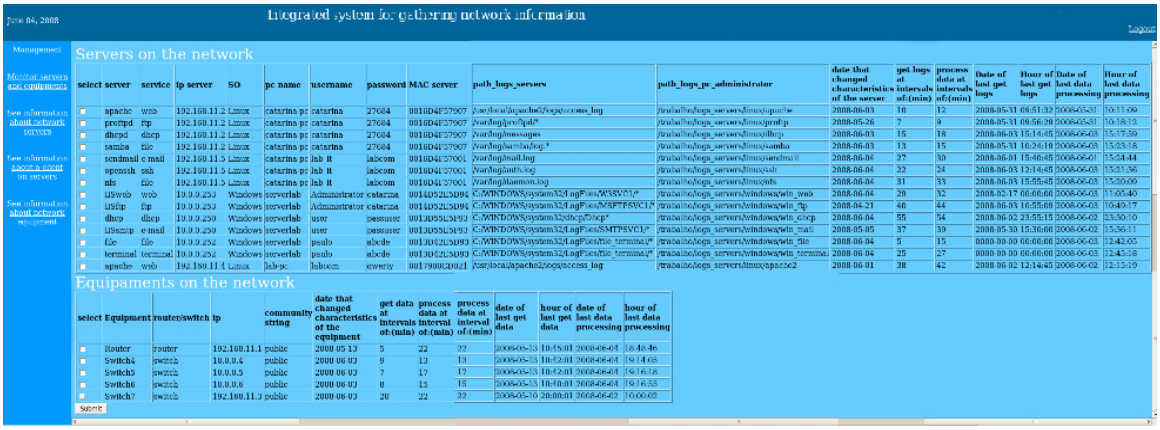
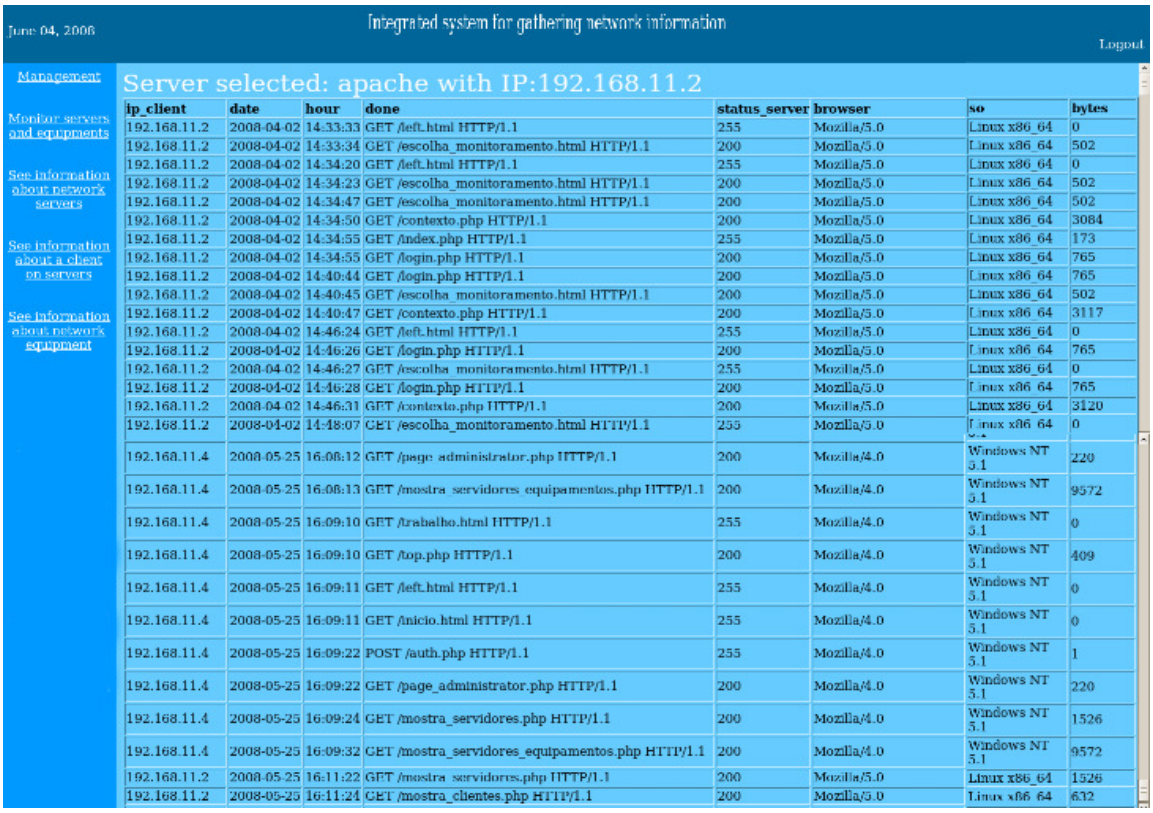


Figura 95 - mostra_servidores_equipamentos.php

Na frame seguinte (que recebe quais os dados seleccionados da frame anterior através de um POST) é apresentada então toda a informação requisitada pelo administrador. No exemplo da

figura seguinte o administrador desejava ver os dados correspondentes ao serviço Apache na máquina com o IP 192.168.11.2 (primeiro Apache da tabela):



ip_client	date	hour	done	status_server	browser	so	bytes
192.168.11.2	2008-04-02	14:33:33	GET /left.html HTTP/1.1	255	Mozilla/5.0	Linux x86_64	0
192.168.11.2	2008-04-02	14:33:34	GET /escolha_monitoramento.html HTTP/1.1	200	Mozilla/5.0	Linux x86_64	502
192.168.11.2	2008-04-02	14:34:20	GET /left.html HTTP/1.1	255	Mozilla/5.0	Linux x86_64	0
192.168.11.2	2008-04-02	14:34:23	GET /escolha_monitoramento.html HTTP/1.1	200	Mozilla/5.0	Linux x86_64	502
192.168.11.2	2008-04-02	14:34:47	GET /escolha_monitoramento.html HTTP/1.1	200	Mozilla/5.0	Linux x86_64	502
192.168.11.2	2008-04-02	14:34:50	GET /contexto.php HTTP/1.1	200	Mozilla/5.0	Linux x86_64	3084
192.168.11.2	2008-04-02	14:34:55	GET /index.php HTTP/1.1	255	Mozilla/5.0	Linux x86_64	173
192.168.11.2	2008-04-02	14:34:55	GET /login.php HTTP/1.1	200	Mozilla/5.0	Linux x86_64	765
192.168.11.2	2008-04-02	14:40:44	GET /login.php HTTP/1.1	200	Mozilla/5.0	Linux x86_64	765
192.168.11.2	2008-04-02	14:40:45	GET /escolha_monitoramento.html HTTP/1.1	200	Mozilla/5.0	Linux x86_64	502
192.168.11.2	2008-04-02	14:40:47	GET /contexto.php HTTP/1.1	200	Mozilla/5.0	Linux x86_64	3117
192.168.11.2	2008-04-02	14:46:24	GET /left.html HTTP/1.1	255	Mozilla/5.0	Linux x86_64	0
192.168.11.2	2008-04-02	14:46:26	GET /login.php HTTP/1.1	200	Mozilla/5.0	Linux x86_64	765
192.168.11.2	2008-04-02	14:46:27	GET /escolha_monitoramento.html HTTP/1.1	255	Mozilla/5.0	Linux x86_64	0
192.168.11.2	2008-04-02	14:46:28	GET /login.php HTTP/1.1	200	Mozilla/5.0	Linux x86_64	765
192.168.11.2	2008-04-02	14:46:31	GET /contexto.php HTTP/1.1	200	Mozilla/5.0	Linux x86_64	3120
192.168.11.2	2008-04-02	14:48:07	GET /escolha_monitoramento.html HTTP/1.1	255	Mozilla/5.0	Linux x86_64	0
192.168.11.4	2008-05-25	16:08:12	GET /page_administrator.php HTTP/1.1	200	Mozilla/4.0	Windows NT 5.1	220
192.168.11.4	2008-05-25	16:08:13	GET /mostra_servidores_equipamentos.php HTTP/1.1	200	Mozilla/4.0	Windows NT 5.1	9572
192.168.11.4	2008-05-25	16:09:10	GET /trabalho.html HTTP/1.1	255	Mozilla/4.0	Windows NT 5.1	0
192.168.11.4	2008-05-25	16:09:10	GET /top.php HTTP/1.1	200	Mozilla/4.0	Windows NT 5.1	409
192.168.11.4	2008-05-25	16:09:11	GET /left.html HTTP/1.1	255	Mozilla/4.0	Windows NT 5.1	0
192.168.11.4	2008-05-25	16:09:11	GET /inicio.html HTTP/1.1	255	Mozilla/4.0	Windows NT 5.1	0
192.168.11.4	2008-05-25	16:09:22	POST /auth.php HTTP/1.1	255	Mozilla/4.0	Windows NT 5.1	1
192.168.11.4	2008-05-25	16:09:22	GET /page_administrator.php HTTP/1.1	200	Mozilla/4.0	Windows NT 5.1	220
192.168.11.4	2008-05-25	16:09:24	GET /mostra_servidores.php HTTP/1.1	200	Mozilla/4.0	Windows NT 5.1	1526
192.168.11.4	2008-05-25	16:09:32	GET /mostra_servidores_equipamentos.php HTTP/1.1	200	Mozilla/4.0	Windows NT 5.1	9572
192.168.11.2	2008-05-25	16:11:22	GET /mostra_servidores.php HTTP/1.1	200	Mozilla/5.0	Linux x86_64	1526
192.168.11.2	2008-05-25	16:11:24	GET /mostra_clientes.php HTTP/1.1	200	Mozilla/5.0	Linux x86_64	632

Figura 96 - ver_resultados.php

Através do **See information about network servers** o administrador pode escolher de qual serviço deseja ver os dados. Escolhe através de comboboxes (que mostram os dados presentes na base de dados sem repetição, utilizando o comando SELECT DISTINCT) o nome do serviço, o sistema operativo, o IP do servidor e o IP do cliente que deseja visualizar no servidor específico. Na figura seguinte apresenta-se esta frame:

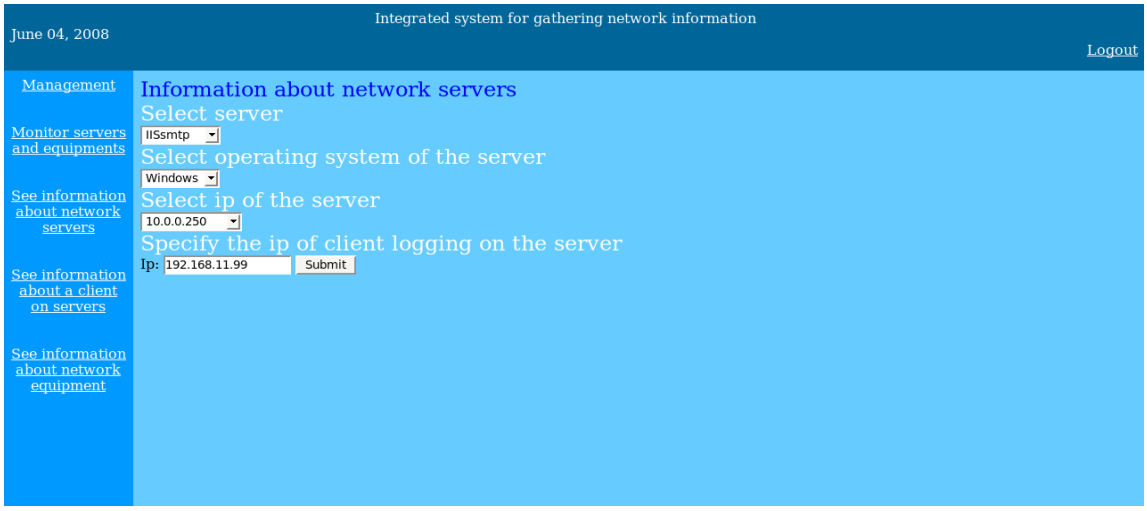


Figura 97 - mostra_servidores.php

Na figura anterior, o administrador escolhe que quer ver o serviço IISsmtp, que está numa máquina com sistema operativo Windows, e cujo IP é 10.0.0.250. Também especifica o IP do cliente como 192.168.11.99. Os dados seleccionados são passados por POST para a frame seguinte.

Primeiro efectua-se uma consulta para verificar se estas informações correspondem a alguma linha da tabela **list_of_servers** com o comando da figura seguinte:

```
$query1 = "SELECT * FROM list_of_servers WHERE serv =  '$server' AND so =
'$sys_oper' AND ip_server = '$server_ip'";
```

Figura 98 - Consulta por serviço, sistema operativo e IP

Em que as variáveis **$server**, **$sys_oper** e **$server_ip** foram as escolhidas nos comboboxes.

Se sim lê-se o identificador do servidor (porque pode haver mais do que uma máquina com o mesmo serviço) e faz-se então outra procura conforme figura seguinte e imprime-se os valores obtidos por este filtro:

```
$query2 = "SELECT * FROM $server WHERE id_server = '$idserver' AND ip =
'$client_ip'";
```

Figura 99 - Consulta no serviço apenas dados com id desejado

Na figura seguinte apresenta-se o resultado desta pesquisa:

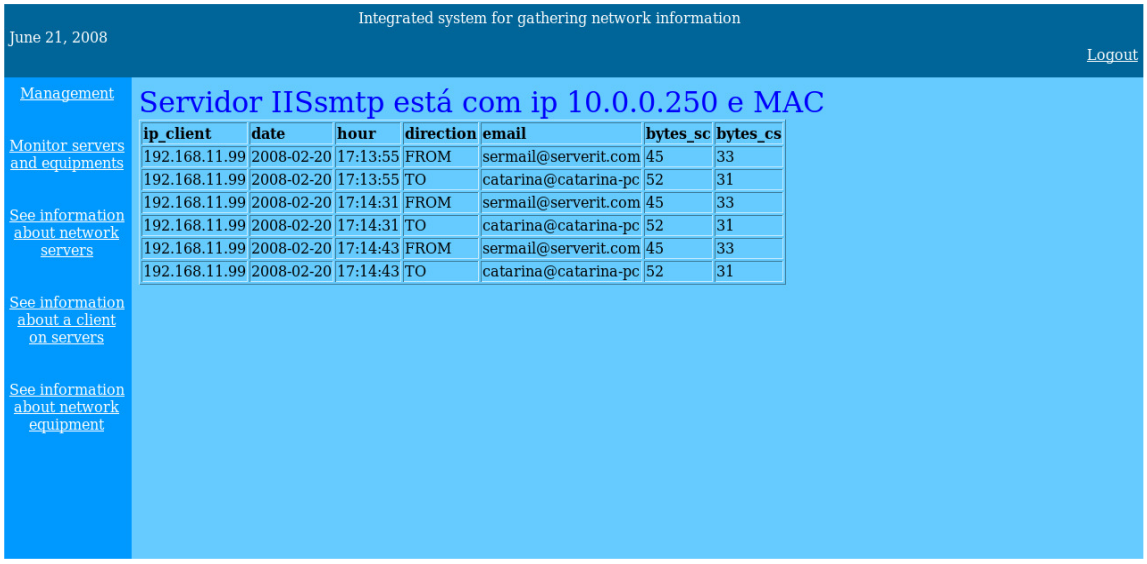


Figura 100 - resultado_servidores.php

Assim, é possível ver a data e hora em que o cliente específico utilizou o serviço e outras informações inportantes.

Através do **See information about a client on servers** é possível verificar todas as ocorrências do cliente na rede após uma determinada data. Introduz-se o IP do cliente e a data de inicio da procura conforme a figura seguinte:

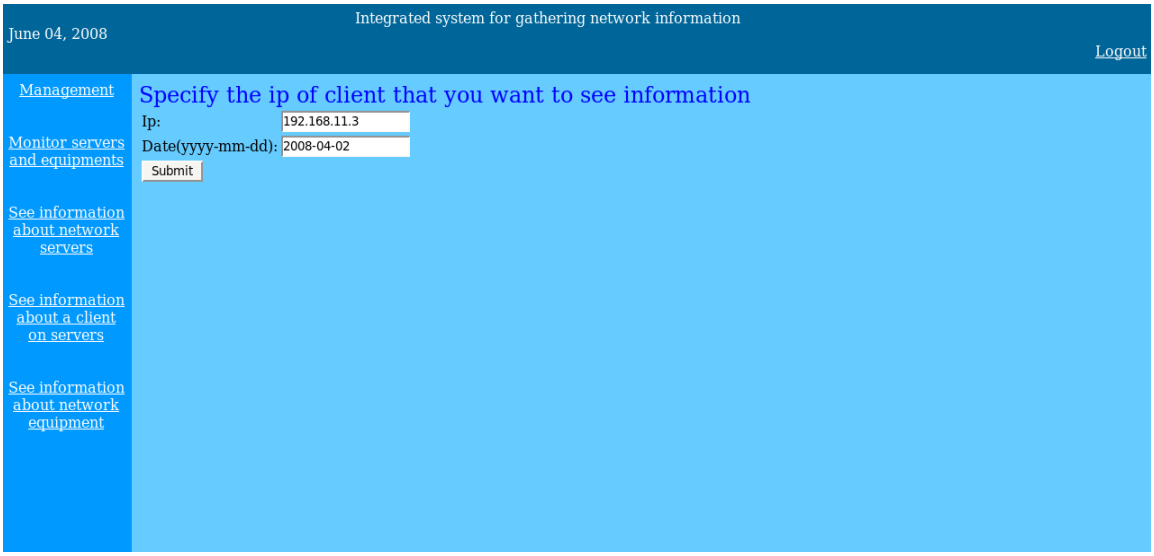


Figura 101 - mostra_clientes.php

Faz-se uma consulta em todas as tabelas dos serviços conforme figura seguinte:

```
$query = "SELECT * FROM $a WHERE ip = '$ipc' AND date >= '$day'";
```

Figura 102 - Consulta cliente

A variável **$a** corresponde ao nome da tabela (varia com um while) e são filtrados os dados com o IP igual a **$ipc** e data maior que **$day.**

Na frame seguinte serão apresentados, conforme o exemplo acima, as informações referentes ao cliente com IP 192.168.11.3 e os serviços que utilizou nas datas superior ou igual ao dia 2 de Abril de 2008.

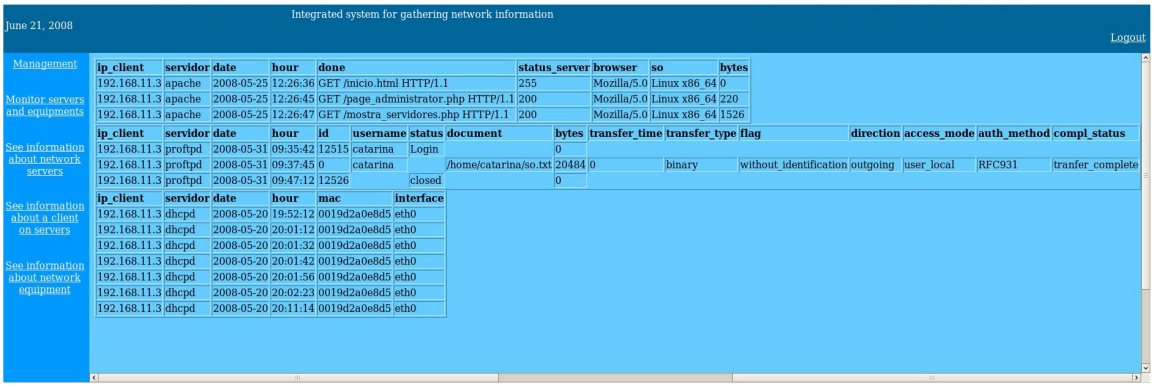


Figura 103 - resultado_clientes.php

Na figura anterior é possível observar que no período determinado este cliente acedeu aos serviços Apache, PROFTPd, DHCPd.

Através do **See information about network equipment** observam-se as características de todos os equipamentos presentes na rede, como por exemplo o nome do equipamento, contacto do administrador, hora e data da última vez que houve a busca dos dados no equipamento respectivo, entre outros. Observam-se na figura seguinte estes equipamentos:

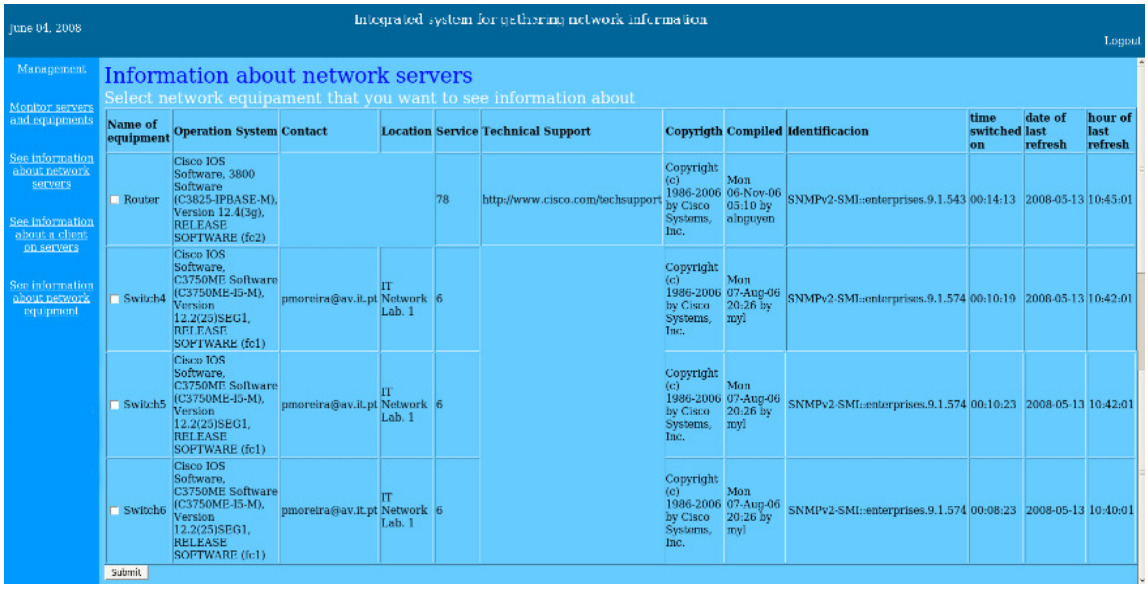


Figura 104 - mostra_equipamentos.php

Através de checkboxes, seleccionam-se os equipamentos dos quais queremos ver as restante informações armazenadas na respectiva base de dados. Na figura seguinte observam-se as tabelas correspondentes ao equipamento de nome **Router**:

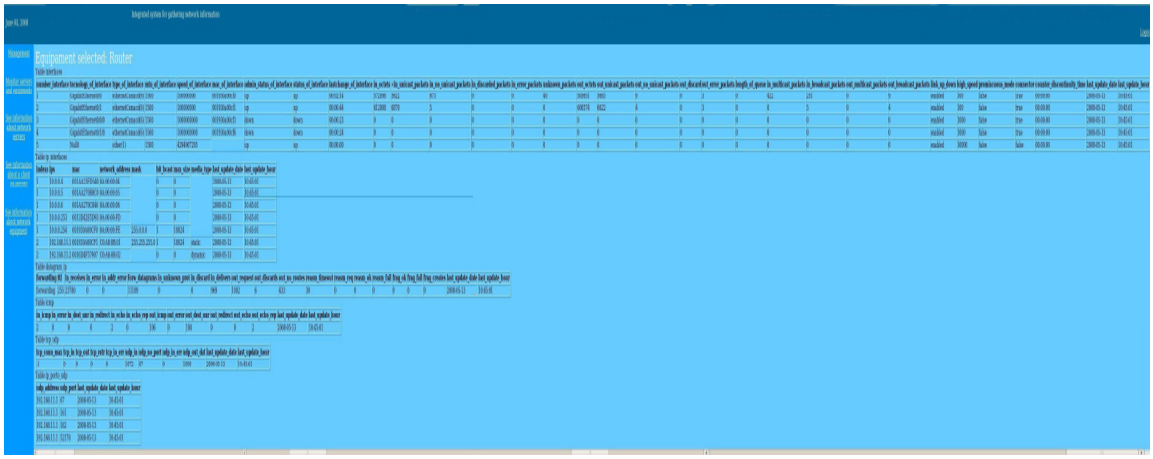


Figura 105 - resultado_equipamentos.php

7.4 Síntese

Neste capítulo foi apresentada a interface *Web* desenvolvida para monitorização deste sistema. Através desta interface o administrador adiciona os equipamentos de rede e serviços que deseja monitorizar e programa o tempo para recolha e processamento da informação recolhida dos mesmos. Mostra-se que também é possível visualizar as informações presentes na rede através de diferentes filtros (por cliente, por servidor, por equipamento, data, entre outros).

8 Conclusões

Neste capítulo são apresentadas as principais conclusões do trabalho efectuado e apresentadas algumas propostas para trabalho futuro.

8.1 Principais conclusões

Nesta dissertação foi apresentado um sistema integrado para recolha de informações de serviços e equipamentos de uma rede. O objectivo principal deste sistema prende-se ao facto da necessidade de haver sistemas que, de forma autónoma, façam todas as etapas de monitorização de uma rede, isto é, desde a recolha até a apresentação dos resultados.

Primeiramente, o administrador deve através da interface *Web* inserir a lista dos serviços e equipamentos que deseja monitorizar na rede. De seguida o sistema, com o auxílio do CRON, irá activar os processos de recolha de informação, através de SCP e SNMP, dos logs dos servidores e informação das MIBS dos equipamentos respectivamente, conforme o tempo programado pelo administrador através da interface *Web*. Também será activado através do CRON o processamento de todos estes dados através de shell scripts. Estes scripts são responsáveis também pela invocação dos scripts SQL que armazenam os dados nas bases de dados criadas. Finalmente, é possível ao administrador visualizar todos estes dados filtrados de diferentes formas, mais uma vez, através da interface *Web*.

8.2 Sugestões para trabalho futuro

- Correlação de todas as informações recolhidas.
- Implementação de um método de detecção automática de anomalias.
- Sistema de alerta quando for detectado qualquer problema na rede.
- Indicar ao administrador possíveis contra-medidas a tomar.

Lista de acrónimos e siglas

Na tabela seguinte são apresentadas as siglas e acrónimos (por ordem alfabética) utilizados neste trabalho:

Siglas/Acrónimos	Significado
ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
CRM	Customer Relation Management
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
ERP	Enterprise Resource Planning
FTP	File Transfer Protocol
HTML	HyperText Mackup Language
HTTP	HyperText Transfer Protocol
ICMP	Internet Control Message Protocol
IDC	International Data Corporation
IOS	Internetwork Operating System
IP	Internet Protocol
ISO	International Organization for Standardization
LAN	Local Area Network
LDR	Lado Direito Rato
MAC	Media Access Control
MD	Managed devices
MIB	Management Information Base
MTA	Mail Transfer Agent
NetBios	Network Basic Input Output System
NFS	Network File System
NMS	Network Management System
OID	Object Identifier
PDU	Prtocol Data Unit
PHP	PHP Hypertext Preprocessor
RDBMS	Relational Database Management System
SMS	Short Message Service
SMI	Structure of Management Information
SMTP	Simple Mail Tranfer Protocol
SNMP	Simple Network Management Protocol
TCP	Transmission Control Protocol
UDP	User Data Protocol
WAN	Wide Area Network

Tabela 41 - Lista de acrónimos e siglas

Bibliografia

Livros

[Converse2004] Converse, T., Park, J., Morgan, C. (2004). *PHP5 and MySQL Bible*. Indianapolis: Wiley Publishing.

[Samba2007] Ts J., Eckestein R., Collier-Brown D. (2003). *Using Samba* (2ª ed). [s. l.]: O'Reilly. Acedido em 13 de Dezembro de 2007, em http://samba.org/samba/docs/using_samba/toc.html

[Kaufman2002] Kaufman, C., Perlman, R., Speciner, M. (2002). *Network security: Private communication in a public world* (2ª ed.). [s. l.]: Prentice Hall

[Welling2001] Welling L., Thomson L. (2001). *PHP and MySQL Web Development*. Indianapolis: Sams.

Artigos e documentos

[ActiveXperts2008] ActiveXperts. Acedido em 26 de Maio de 2008, em http://www.activexperts.com/

[Alertbot2008] AlertBot. Acedido em 26 de Maio de 2008, em http://www.alertbot.com

[Apache2008] Apache. Acedido em 10 de Setembro de 2007, em http://httpd.apache.org

[ApacheLogs2008] *Log Files*. Acedido em 10 de Setembro de 2007, em http://httpd.apache.org/docs/1.3/logs.html

[ASN2008] *Introduction to ASN.1*. Acedido em 20 de Fevereiro de 2008, em http://asn1.elibel.tm.fr/en/introduction/index.htm

[CISCOSWITCH2008] *MIBs supported by Metro Ethernet Catalyst 3750ME serie*. Acedido em 18 de Fevereiro de 2008, em ftp://ftp.cisco.com/pub/mibs/supportlists/cat3750me/cat3750me-supportlist.html

[CISCOROUTER2008] *MIBs supported by Router 3800 series.* Acedido em 18 de Fevereiro de 2008, em http://www.cisco.com/en/US/prod/collateral/routers/ps5854/product_data_sheet0900aecd80581fe6_ps5855_Products_Data_Sheet.html

[CISCO2008] *Network management basics*. Acedido em 10 de Novembro de 2007, em http://www.cisco.com/en/US/docs/internetworking/technology/handbook/NM-Basics.html

[Crosstecsecurity2008] Activeworx. Acedido em 25 de Maio de 2008, em http://www.crosstecsecurity.com/

[Domingues2004] Domingues M. (2004). *Desenvolvimentos avançados em redes: Monitorização e Prevenção*. Acedido em 15 de Março de 2008, em http://www.fccn.pt/files/documents/D2.08.PDF

[DOStoUNIX2007] Conversão de formatos DOS to UNIX. Acedido em 01 de Março de 2008, em http://www.dicas-l.com.br/dicas-l/20070703.php

[Expect2008] Expect. Acedido em 26 de Maio de 2008, em http://expect.nist.gov/

[GFI2008] GFI. Acedido em 24 de Maio de 2008, em http://www.gfi.comdido

[GrokEVT2008] GrokEVT Home. Acedido em 15 de Abril de 2008, em http://projects.sentinelchicken.org/grokevt/

[HealthMonitor2008] Health Monitor. Acedido em 26 de Maio de 2008, em http://www.health-monitor.com/

[HPOpenView2008] HP OpenView. Acedido em 26 de Maio de 2008, em http://www.openview.hp.com/

[IDC2008] IDC (2008). *The Role of Linux Servers and Commercial Workloads*. Acedido em 15 de Abril de 2008, em http://www.linux-foundation.org/publications/IDC_Workloads.pdf

[IISFAQ2008] Article 435. Acedido em 26 de Maio de 2008, em http://www.iisfaq.com/default.aspx?View=A435

[Lire2008] Lire. Acedido em 15 de Janeiro de 2008, em http://www.logreport.org/lire.html

[Microsoft2008] *Monitoramento de segurança e detecção de ataques*. Acedido em 10 de Janeiro de 2008, em http://www.microsoft.com/brasil/technet/security/midsizebusiness/topics/serversecurity/attackdetection.mspx

[MySQL2008] MySQL 5.1 Reference Manual. Acedido de 10 de Fevereiro a 20 de Maio de 2008, em http://dev.mysql.com/doc/refman/5.1/en/index.html

[Nagios2008] Nagios. Acedido em 28 de Maio de 2008, em http://www.nagios.org

[NetSNMP2008] Net-SNMP. Acedido em 10 de Março de 2008, em http://net-snmp.sourceforge.net/

[PHP2008] PHP. Acedido em 28 de Março de 2008, em http://www.php.net/

[RFC3164 2008] *RFC3164*. Acedido em 31 de Maio de 2008, em http://tools.ietf.org/html/rfc3164

[Xpolog2008] XPLG. Acedido em 26 de Maio de 2008, em http://xpolog.com/

[Webmin2008] Webmin. Acedido em 06 de Outubro de 2007, em http://www.webmin.com/

Anexos

Anexo I - Shell e seus scripts

Shell é um interpretador de comandos do Linux, Unix em geral. A linguagem shell é interpretada, não sendo necessário compilar o código. Dos vários Shell existentes, o Bourne Shell, o Korn Shell, o C Shell e a Bourne Again Shell (bash), este último tem sido o mais utilizado.

O Bourne Shell (sh) foi escrito por Stephen Bourne um pesquisador da AT&T Bell Labs, e foi durante muito tempo o shell default e único nos sistemas operativos UNIX.

O Korn Shell (ksh) foi desenvolvido por David Korn, também da Bell Labs e é um subset do sh.

O C Shell (csh) foi desenvolvido por Billy Joy da Berkley University é o Shell mais utilizado em ambientes *BSD e Xenix. Estruturação dos comandos é muito similar à da linguagem C.

O utilizado neste trabalho é o Bourne Again Shell (bash). É compatível com o sh original e incorpora os melhores recursos do Korn Shell e C Shell. Ele segue o padrão IEEE Posix shell e inicialmente foi escrito para ser o shell padrão do sistema GNU da Free Software Fundation. Ele tem uma portabilidade boa, existem implementações para sistemas Windows, MacOsX e Linux e BSDs. Para utilizadores de Windows ele pode ser usado através do projeto Cygwin.

Os comandos podem ser enviados de duas formas para o interpretador:

- Interactiva: comandos são digitados na linha de comandos e passados ao interpretador de comandos um a um.
- Não interactiva: comandos são passados na forma de scripts, os shell scripts. Não são mais do que uma sequência de comandos interpretados um após o outro pelo sistema operativo.

A primeira linha dum ficheiro script indica o nome do shell que irá interpretar o script. Neste trabalho, como utilizaremos o *bash*, temos na primeira linha a instrução **#!/bin/bash.**

É necessário também tornar o script executável. Para isso com permissões de sudo usa-se o comando **chmod +x nome_script.**

Para chamar o script usam-se os comandos:

Comando	Condição
./nome_script	Se o script estiver no directório corrente
$path/nome_script (path=caminho completo desde a raiz)	Se o script não estiver no directório corrente

Tabela 42 - Comandos executar scripts

Passagem de parâmetros:

Passam-se os argumentos directamente na linha de comandos: chama-se o script da mesma forma especificada na tabela anterior e com os parâmetros escritos à frente. Dentro do script os argumentos recebidos são pela ordem $1, $2, $3…e assim por diante. Se o número de parâmetros for maior do que nove utiliza-se chaves na notação: ${10}, ${11} e assim por diante. O valor $0 indica o nome do script.

Comentários:

São feitos para tornar o código mais claro e fácil de entender. Deve-se colocar o carácter # no inicio da linha.

Variáveis:

A atribuição é feita com a sintaxe: nome_variavel=valor. Não são permitidos espaços nem antes nem depois do carácter =. Apenas caracteres alfanuméricos podem ser utilizados como identificadores válidos de variáveis e o nome da variável só pode começar por letras ou *underline*. Os valores do tipo string que contenham espaços devem ser especificados entre aspas.

Para uma variável ser referenciada usa-se um $. No caso $nome_variavel é o valor do nome_variavel.

Arrays:

É permitido apenas arrays com uma dimensão (vector). Os elementos podem ser definidos com a sintaxe: variável[indice]=valor ou no caso de definirmos o array todo este deve estar entre ().Para obter o valor de um elemento do array utilize-se a sintaxe ${variável[indice]}.

Pipes:

Juntam-se os comandos permitindo que o resultado de um comando seja passado para outro comando. Representado por "|" entre os comandos.

Comandos:

cat $path : exibe o conteúdo do ficheiro indicado. Opções:

-n: numera as linhas.

-b: numera as linhas, excepto as que estão em branco.

-s: não exibe mais de uma linha em branco simultaneamente.

cat $path $path1 : exibe o conteúdo dos dois ficheiros concatenados.

head -n x $path : exibe o conteúdo das primeiras x linhas do ficheiro. Se x não for especificado mostra as 10 primeiras linhas.

tail -n x $path : exibe o conteúdo das últimas x linhas do ficheiro. Se x não for especificado mostra as 10 últimas linhas.

cut : o conteúdo é dividido em campos de acordo com o padrão passado. Opções:

-cx-y : exibe o conteúdo das colunas de x a y.

-d"x": divide o conteúdo (especificar entre aspas ou apóstrofe) através do delimitador x.

-fcampo : especifica qual o campo desejado.

awk '/padrao/ {acção}': Podem ser feitas inúmeros processamentos com este comando. Neste trabalho utilizou-se o awk apenas com o padrão seguinte: **awk '{print $x}'**. Este da-nos o campo que está na posição x.

echo "x": imprime a frase x na console.

grep x : exibe o conteúdo apenas das linhas que contém a palavra x. Opções:

-v: exclui uma expressão no resultado.

-i: ignora a caixa de letra, sem fazer distinções entre maiúsculas e minúsculas.

-r: opera recursivamente, isto é, procura no directório local e em seus sub-directórios.

-s: não exibe mensagens de erro.

-A x: exibe o resultado procurado e as x linhas seguintes.

-B x: exibe o resultado procurado e as x linhas anteriores.

egrep x : exibe o conteúdo apenas das linhas que não contêm a palavra x.

sed : edita texto. Pode ser usado para substituir uma string por outra por exemplo, adicionar linhas quando encontra padrão, dentre outros. Opções:

s: Usado na substituição. Funciona da seguinte forma: um s precedido de um caracter separador, precedido da expressão a ser substituída, precedido da expressão que irá substituir, precedido da linha.

wc ficheiro: conta às linhas, palavras e caracteres do ficheiro. As opções mais usadas são:

-c: mostra a número de caracteres.

-l: mostra a número de linhas.

-w: mostra a número de palavras.

-L: mostra o comprimento da linha mais longa.

date: mostra a data e hora actual. Comando **date '+%d%m%y'** mostra a data no formato ddmmaa.

let: usado como contador crescente ou decrescente, na atribuição de um valor ou numa operação aritmética.

read: realiza a leitura de uma variável digitada no teclado. Exemplo:

```
echo “Digite o seu nome”
read NOME
echo $NOME
```

Figura 106 - Comando read

sleep n : Para parar a execução do script por n seg.

Expressões regulares:

Servem para especificar padrões que pretendemos usar:

Comando	Significado	Exemplo
^	que começa com	cat ficheiro | grep ^void Procura linhas que comecem com void
$	que termina com	cat ficheiro | grep ^void Procura no ficheiro linhas que terminem com void
[…]	que contém qualquer uma das letras	cat ficheiro | grep "com[ea]" Procura no ficheiro linhas com a palavra come ou coma.
[^…]	que não contém as letras	cat ficheiro | grep "com[ea]" Procura no ficheiro linhas sem a palavra come ou coma.
palavra1|palavra2	que contém palavra1 ou palavra2	cat ficheiro | grep "hojelamanha" Procura no ficheiro as linhas com a palavra hoje ou amanha.

Tabela 43 - Expressões regulares shell scripts

Redireccionamento:

Comando	Significado
>	redirecciona o resultado para outro ficheiro
>>	redirecciona o resultado para o fim do ficheiro (não substitui e sim adiciona)
<	redirecciona a entrada padrão usando um ficheiro
2>	redirecciona a saída de erro para ficheiro.

2>>	redirecciona a saída de erro para o fim do ficheiro
>&2	redirecciona a saída padrão para saída de erro
2>&1	redirecciona saída de erro para saída padrão

Tabela 44 - Comandos redireccionamento shell scripts

Metacaracteres:

São caracteres especiais. Utiliza-se "\" antes dos metacaracteres com a função de indicar ao interpretador que se deve interpretar o caracter literalmente sem nenhum tipo de expansão.

Operadores:

aritméticos:

comando	significado
+	soma
-	subtracção
*	multiplicação
/	divisão
%	modulo
**	exponencial

Tabela 45 - Operadores aritméticos

Uma expressão aritmética em bash é chamada por parêntesis duplos com o uso dos operadores da tabela acima.

lógicos:

O shell não avalia as duas expressões pois teria custo de processamento desnecessário. Avalia apenas consoante os casos seguintes:

comandox && comandoy: depende se houve sucesso na operação anterior. O comandoy só é executado se o comandox tiver sucesso.

comandox || comandoy: depende se houve falha na operação anterior. O comandoy só é executado se o comandox não tiver sucesso.

Comparação inteira:

comando	significado
-eq	Igual
-ne	Diferente
-gt	Maior que
-ge	Maior ou igual
-lt	Menor que
-le	Menor ou igual

Tabela 46 - Comparação inteira

Comparação de strings:

comando	significado
=	Igual
!=	Diferente
<	Menor que
>	Maior que
-z	String nula (tamanho = 0)
-n	String não nula

Tabela 47 - Comparação de strings

Expressões com ficheiros:

comando	significado
-e	se o arquivo existe
-d	se o arquivo é um directório
-s	se o arquivo não é vazio
-h	se o arquivo é um link simbólico
-r	se o arquivo pode ser lido pelo usuário
-w	se o arquivo pode ser escrito pelo usuário
-x	se o arquivo pode ser executado pelo usuário
-O	se pertence ao usuário
-G	se pertence ao grupo do usuário
-N	se foi modificado desde a última leitura

Tabela 48 - Expressões com ficheiros

Estruturas de controlo de decisão:

Estrutura if	Estrutura case
if [condição1] then comandos se condição1 for verdadeira elif [condição2] then comandos se condição2 for verdadeira else comandos se nenhum das condições acima for verdadeira fi	case variável in opção1) comando1 ;; opção2) comando 2 ;; *) comando_default ;; esac

Tabela 49 - Estruturas de controlo de decisão

Estruturas de repetição:

Estrutura for: a variável contadora assume o valor de uma lista a cada interacção e não necessariamente é uma estrutura numérica. Sua variável de controlo pode assumir strings. Dois formatos utilizados são os da tabela seguinte:

Estrutura for	
for x in lista_de_valores; do comandos done	for ((x=y ; x<z; x++)); do comandos done

Tabela 50 - Estrutura for

Quando temos um ciclo for que caminha por valores de um array é útil sabermos o número de elementos do array usado. Logo pode ser feito:

```
nomes=( João Maria Ana Sara )
for (( x=0 ; x<${#nomes[*]}; x++ ));
do
        comandos
done
```

Figura 107 - Nova estrutura for

Na estrutura acima o ${#nomes[*]} representa o numero de elementos no array nomes, que é igual a quatro.

Estrutura while: no cabeçalho está a condição de execução e repete os comandos enquanto a condição é verdadeira. Sintaxe:

```
while [condição]
do
        comandos
done
```

Figura 108 - While shell scripts

Estrutura until: no cabeçalho está a condição de paragem. Sintaxe:

```
until [condição]
do
        comandos
done
```

Figura 109 - Until shell scripts

Função:

Torna possível organizar o código em blocos tornando possível a não repetição de código. Pode ser feito nos dois formatos apresentados na tabela a seguir:

function nome_da_função{
comandos
}
	nome_da_função( ){
	comandos
	}

Tabela 51 - Formatos função

Os argumentos das funções são passados de forma idêntica de que num script: $1 é o primeiro argumento, $2 o segundo e assim sucessivamente.

Anexo II - Instalação e Configuração dos Serviços em Linux

Apache

A melhor solução para instalar este serviço é fazer o *download* da versão mais recente na página *Web* do Apache [Apache2008] ao invés de utilizarmos o comando **sudo apt-get install apache** pois, é importante termos o ficheiro **configure** da pasta de instalação e através apenas do comando referido não ficaria disponível. Passos em detalhe:

```
tar -zxvf httpd-2.2.4.tar.gz
cd httpd-2.2.4
./configure --
prefix=/usr/local/apache2
make
sudo make install
```

Figura 110 - Comandos instalação Apache

A quarta linha da figura anterior indica o *filesystem path.*

O ficheiro de configuração fica em **/usr/local/apache2/conf/httpd.conf**. Neste ficheiro é possível configurar o porto que o servidor escuta, por defeito é o 80. Também é possível configurar um endereço IP fixo (se não tiver configurado DNS no servidor), o usuário e grupo, e o formato do ficheiro de log, modificado segundo sub-capítulo 4.2.1.1. Posteriormente na instalação do PHP foram feitas outras mudanças no ficheiro de configuração deste servidor e estão descritas no Anexo VII.

Para iniciar o servidor utiliza-se o comando **sudo /usr/local/apache2/bin/apachectl start**, e para parar o servidor utiliza-se o comando **sudo /usr/local/apache2/bin/apachectl stop**.

ProFTPD

Sua instalação foi feita com o comando **sudo apt-get install proftpd**.

O ficheiro de configuração localiza-se em **/etc/proftpd/proftpd.conf**. Este é muito semelhante ao ficheiro de configuração do Apache, no qual também é possível configurar o porto que o servidor escuta, um endereço IP fixo (se não tiver configurado DNS no servidor), o usuário e grupo, o número máximo de clientes.

Apesar desta versão apresentar suporte IPv6 (desde a versão 1.2.9rc2 o protocolo é suportado), retirou-se do ficheiro de configuração esta opção: **UseIPv6 off**

Antes de se modificarem as configurações deve-se sempre parar o servidor através do comando **sudo /etc/init.d/proftpd stop** e depois de efectuadas, iniciar com o comando **sudo /etc/init.d/proftpd start**.

DHCPd

Num terminal, digita-se o comando **sudo apt-get install dhcp3-server** para instalar o dhcpd. Para configurar o servidor:

- Para-se o serviço com o comando: **/etc/init.d/dhcp3-server stop**
- Edita-se o ficheiro de configuração que esta em **/etc/dhcp3/dhcpd.conf**. No âmbito deste trabalho adicionou-se a este ficheiro as seguintes linhas:

```
default-lease-time 600;
max-lease-time 7200;
subnet 192.168.11.0 netmask 255.255.255.0 {
range 192.168.11.1 192.168.11.100;
option routers 192.168.11.1;
option domain-name-servers 192.168.11.1,213.228.128.6,213.228.128.5;
option broadcast-address 192.168.11.255;
host catarinahome {
hardware ethernet 08:00:27:9B:15:E2;
fixed-address 192.168.11.101;
}
host CATA {
hardware ethernet 00:30:4F:0B:03:40;
fixed-address 192.168.11.103;
}
}
```

Figura 111 - Configuração Dhcpd

As duas primeiras linhas indicam que os endereços serão atribuídos por um período de 600 segundos, isto se o cliente não especificar outro intervalo. Caso contrário, o tempo máximo permitido de atribuição será de 7200 segundos. A terceira linha indica a rede e máscara. A quarta linha indica a gama de endereços *address poll*. A quinta, sexta e sétima linha indicam respectivamente o router, servidores DNS e endereço de difusão. As restantes linhas são atribuições fixas, em que devemos indicar o nome da máquina, o MAC da interface e o endereço IP que queremos que lhe seja atribuído.

- Edita-se o ficheiro **/etc/default/dhcp3-server** para definir as interfaces de rede que o DHCP deve escutar. Neste trabalho a interface utilizada era a *eth0* logo, o ficheiro apresentava a seguinte linha:

```
INTERFACES="eth0"
```

Figura 112 - Interface DHCP

- Reinicializa-se o serviço DHCP com o comando: **/etc/init.d/dhcp3-server restart**

OpenSSH

Num terminal, digite o comando **sudo apt-get install ssh** para instalar o servidor e cliente (pacote inclui os dois). Inicia-se o servidor com o comando **/etc/init.d/ssh start**. O ficheiro de configuração do servidor esta no ficheiro **/etc/ssh/sshd_config** e do cliente no **/etc/ssh/ssh_config**.

No âmbito deste trabalho não foram realizadas alterações no ficheiro de configuração.

Samba

Para instalar o serviço digita-se na linha de comandos: **sudo apt-get install samba**.

Ficheiro de configuração esta em **/etc/samba/smb.conf.** Este ficheiro é estruturado da seguinte forma: os parâmetros de configuração são agrupados em secções e cada uma é identificada por um nome entre parêntesis rectos. Na tabela seguinte tem-se a descrição de três exemplos de secções pré-definidas, mas existem outras:

Secção	Descrição
[global]	contém configurações que afectam o Samba
[homes]	configurações do directório home para cada usuário
[printers]	configurações que controlam impressoras compartilhadas

Tabela 52 - Exemplo secções smb.conf

As configurações de utilizadores e *passwords* foram configurados através do Webmin.

Nfs

Para instalação no servidor usar comando:

```
sudo apt-get install nfs-kernel-server nfs-common portmap
```

Figura 113 - Instalar nfs

Deve-se reconfigurar portmap para não ouvir a *loopback* interface com os comandos:

```
sudo dpkg-reconfigure portmap
sudo /etc/init.d/portmap restart
```

Figura 114 - Configurar portmap

Devem-se definir os directórios que serão compartilhados, quem tem a permissão e o nível dessa permissão com o comando: **sudo nano /etc/exports.** Deve-se seguir o formato seguinte:

Comando	**Descrição**
path rede/nºde bits1da máscara(rw)	os clientes na gama de ips especificada tem permissão de leitura e escrita dos ficheiros do path especificado.
path rede/nºde bits1da máscara(ro)	os clientes na gama de ips especificada tem permissão apenas de leitura dos ficheiros do path especificado.

Tabela 53 - Permissões NFS

No âmbito deste trabalho o ficheiro esta configurado de forma a que as máquinas com endereço ip entre 192.168.11.1 e 192.168.11.254 tenham permissões de acesso, leitura e escrita da pasta capturas que no servidor esta com o caminho indicado:

```
/home/catarina/Desktop/capturas 192.168.11.0/24(rw)
```

Figura 115 - Exemplo configuração

No fim deve-se salvar e reiniciar o servidor com comando **sudo /etc/init.d/nfs-kernel-server restart** e para exportar a configuração fazer **sudo exportfs -a**

Para instalação no cliente usar comando **sudo apt-get install portmap nfs-common**. Para montar automaticamente utiliza-se o comando **sudo mkdir /mnt/files.** Editam-se as configurações usando **gksudo gedit /etc/fstab** e adicionando comando com o formato:

```
ip_servidor:path_servidor path_cliente nfs size=8192,wsize=8192,timeo=14,intr 0 0
```

Figura 116 - Comando NFS

Neste trabalho encontra-se configurado da seguinte forma:

```
192.168.11.2:/home/catarina/Desktop/capturas /mnt/files nfs
rsize=8192,wsize=8192,timeo=14,intr 0 0
```

Figura 117 - Exemplo configuração NFS

Para testar a nova configuração utiliza-se comando **sudo mount -a**

Sendmail

Instalou-se o sendmail através do repositório de aplicações para Linux. As contas de e-mail seguem a notação user@localhost e para envio e recepção de e-mails de teste utilizou-se o Webmin.

Anexo III - Instalação e Configuração dos Serviços em Windows

IISWEB

Para instalação deste serviço utilizou-se o Manage Your Server, aplicação disponível no Window Server. Opção Add or Remove a role e escolhe-se Application Server (IIS).

Para configura-lo, através do Manage Server, faz-se Manage this application server → Web Sites (LadoDireiroRato"LDR") →New Web Site. Escolhe-se o ip e o porto que o servidor irá escutar, home directory, as permissões de acesso (read, run scripts, execute, write, browse).

IISFTP

Instalação é feita quando instalado o IIS (ver sub-capítulo 4.3.1.1). Para configura-lo, através do Manage Server, faz-se Manage this application server → FTP Sites (LDR) →New FTP Site. Escolhe-se o ip e porto que o servidor irá escutar, o tipo de acesso e escolhe-se o directório ou ficheiro que será compartilhado na ligação FTP. O tipo de acesso pode ser uma das três opções seguintes:

- Não isolar utilizadores: os utilizadores podem aceder à pastas de outros utilizadores.
- Isolar utilizadores: os utilizadores não podem aceder à pastas de outros utilizadores, tendo acesso apenas a sua pasta e sub-pastas dentro desta.
- Isolar utilizadores utilizando o Active Directory: os utilizadores não podem aceder à pastas de outros utilizadores, tendo sua configuração definida pelo Active Directory.

Por fim configuram-se as permissões de acesso (leitura apenas ou leitura e escrita).

Após estas configurações podem-se, através da aba FTP, modificar algumas destas configurações e configurar outras, tais como, o formato do ficheiro de log e o local que ficará armazenado. Visualiza-se também todas as conexões activas, se define se as conexões anónimas são permitidas ou não, entre outras.

DHCP

Por defeito não vem instalado no WinServer e utiliza-se o Manage your Server para instalação. Opção Add or Remove a role e escolhe-se DHCP server.

Para configuração escolhe-se Manage this DHCP server e configura-se a gama de aluguer de endereços, um nome para esta reserva e o tempo de aluguer. É possível também excluir alguns endereços de dentro da gama da reserva, configurar um default gateway para a rede, e o domínio de

servidor DNS. Podem-se fazer reservas de endereços para um cliente. Para isso, basta saber o endereço MAC da interface no cliente.

IISSMTP

Instalação é feita quando instalado o IIS. A seguir, para configurar faz-se Manage this application server → Default SMTP Virtual Server (LDR) →New Virtual Server. Configura-se um nome, ip, home directory, o domínio.

Terminal

Para instalação deste serviço também utilizou-se o Manage Your Server, opção Add or Remove a role e escolhe-se Terminal server.

A seguir, para configurar faz-se Open Terminal Services Configuration →Connections (LDR) →Create New Connection. Escolhe-se o protocolo (RDP), o nível de encriptação, o nome e tipo do protocolo de transporte (tcp), a interface de rede do computador, e o número limite de conexões.

Através do Open Terminal Services Manager é possível observar quantas conexões estão activas.

File

Para instalação deste serviço, também utilizou-se o Manage Your Server, opção Add or Remove a role e escolhe-se File server.

A seguir, para configurar faz-se Manage this file server e no Shared Folders→ Shares (LDR) →New Share. Definem-se as pastas que se quer partilhar, escolhe-se o tipo de acesso (read-only, administrador com *full access* e outros com *read-only*, administrador com *full access* e outros com *read-write*).

Anexo IV- MIBs suportadas pelos equipamentos de rede

As seguintes MIBs são suportadas pelos equipamentos utilizados neste trabalho:

Switch Cisco Catalyst 3750 Metro [CISCOSWITCH2008]	**Router Cisco 3825** [CISCOROUTER2008]
CISCO-CDP-MIB	ATM-MIB
CISCO-CLASS-BASED-QOS-MIB	CISCO-AAL5-MIB
CISCO-CLUSTER-MIB	CISCO-ATM-PVCTRAP-EXTN-MIB
CISCO-CONFIG-MIB	CISCO-BUS-MIB
CISCO-CONFIG-MIB	CISCO-ENT-ASSET-MIB
CISCO-DHCP-SNOOPING-MIB	CISCO-ENTITY-FRU-CONTROL-MIB
CISCO-ENTITY-FRU-CONTROL-MIB	CISCO-IETF-ATM2-PVCTRAP-MIB
CISCO-ENVMON-MIB	CISCO-IETF-ATM2-PVCTRAP-EXTN-MIB
CISCO-FLASH-MIB	CISCO-LECS-MIB
CISCO-FTP-CLIENT-MIB	CISCO-LES-MIB
CISCO-HSRP-MIB	ENTITY-MIB
CISCO-HSRP-EXT-MIB	ETHERLIKE-MIB
CISCO-IGMP-FILTER-MIB	IF-MIB
CISCO-IMAGE-MIB	LAN -EMULATION-CLIENT-MIB
CISCO-L2L3-INTERFACE-MIB	OLD-CISCO-CHASSIS-MIB
CISCO-MAC-NOTIFICATION-MIB	RFC1213-MIB
CISCO-MEMORY-MIB	RFC1253-MIB
CISCO-PAGP-MIB	RMON-MIB
CISCO-PING-MIB	SONET-MIB
CISCO-PORT-QOS-MIB	
CISCO-PROCESS-MIB	
CISCO-RTTMON-MIB	
CISCO-STACKMATER-MIB	
CISCO-STP-EXTENSIONS-MIB	
CISCO-SYSLOG-MIB	
CISCO-TCP-MIB	
CISCO-UDLDP-MIB	
CISCO-VLAN-IFTABLE-RELATIONSHIP-MIB	
CISCO-VLAN-MEMBERSHIP-MIB	
CISCO-VTP-MIB	
ENTITY-MIB	
ETHERLIKE-MIB	
EXTENDED-BRIDGE-MIB	
IEEE8021-PAE-MIB	
IEEE8023-LACP-MIB	
IF-MIB	
IGMP-MIB	
IPMROUTE-MIB	
MPLS-LDP-MIB	
MPLS-LSR-MIB	
MPLS-VPN-MIB	
OLD-CISCO-CHASSIS-MIB	
OLD-CISCO-MIB	
OLD-CISCO-FLASH-MIB	
OLD-CISCO-INTERFACES-MIB	
OLD-CISCO-IP-MIB	
OLD-CISCO-SYS-MIB	
OLD-CISCO-TCP-MIB	
OLD-CISCO-TS-MIB	
PIM-MIB	

RFC1213-MIB	
RFC1253-MIB	
RMON-MIB	
RMON2-MIB	
SNMP-FRAMEWORK-MIB	
SNMP-MPD-MIB	
SNMP-NOTIFICATION-MIB	
SNMP-TARGET-MIB	
SNMP2-MIB	
TCP-MIB	
UDP-MIB	

Tabela 54 - MIBS suportadas nos equipamentos de rede utilizados

Anexo V – Opções Net-SNMP

```
USAGE: snmpwalk [OPTIONS] AGENT [OID]

  Version:  5.2.3
  Web:      http://www.net-snmp.org/
  Email:    net-snmp-coders@lists.sourceforge.net
OPTIONS:
  -h, --help           display this help message
  -H                   display configuration file directives understood
  -v 1|2c|3            specifies SNMP version to use
  -V, --version        display package version number
SNMP Version 1 or 2c specific
  -c COMMUNITY         set the community string
SNMP Version 3 specific
  -a PROTOCOL          set authentication protocol (MD5|SHA)
  -A PASSPHRASE        set authentication protocol pass phrase
  -e ENGINE-ID         set security engine ID (e.g. 800000020109840301)
  -E ENGINE-ID         set context engine ID (e.g. 800000020109840301)
  -l LEVEL             set security level (noAuthNoPriv|authNoPriv|authPriv)
  -n CONTEXT           set context name (e.g. bridge1)
  -u USER-NAME         set security name (e.g. bert)
  -x PROTOCOL          set privacy protocol (DES)
  -X PASSPHRASE        set privacy protocol pass phrase
  -Z BOOTS,TIME        set destination engine boots/time
General communication options
  -r RETRIES           set the number of retries
  -t TIMEOUT           set the request timeout (in seconds)
Debugging
  -d                   dump input/output packets in hexadecimal
  -D TOKEN[,...]       turn on debugging output for the specified TOKENs
                       (ALL gives extremely verbose debugging output)
General options
  -m MIB[:...]         load given list of MIBs (ALL loads everything)
  -M DIR[:...]         look in given list of directories for MIBs
  -P MIBOPTS           Toggle various defaults controlling MIB parsing:
                       u:  allow the use of underlines in MIB symbols
                       c:  disallow the use of "--" to terminate comments
                       d:  save the DESCRIPTIONs of the MIB objects
                       e:  disable errors when MIB symbols conflict
                       w:  enable warnings when MIB symbols conflict
                       W:  enable detailed warnings when MIB symbols conflict
                       R:  replace MIB symbols from latest module
  -O OUTOPTS           Toggle various defaults controlling output display:
                       0:  print leading 0 for single-digit hex characters
                       a:  print all strings in ascii format
                       b:  do not break OID indexes down
                       e:  print enums numerically
                       E:  escape quotes in string indices
                       f:  print full OIDs on output
                       n:  print OIDs numerically
                       q:  quick print for easier parsing
                       Q:  quick print with equal-signs
                       s:  print only last symbolic element of OID
                       S:  print MIB module-id plus last element
                       t:  print timeticks unparsed as numeric integers
                       T:  print human-readable text along with hex strings
                       u:  print OIDs using UCD-style prefix suppression
```

```
                U:  don't print units
                v:  print values only (not OID = value)
                x:  print all strings in hex format
                X:  extended index format
-I INOPTS           Toggle various defaults controlling input parsing:
                b:  do best/regex matching to find a MIB node
                h:  don't apply DISPLAY-HINTs
                r:  do not check values for range/type legality
                R:  do random access to OID labels
                u:  top-level OIDs must have '.' prefix (UCD-style)
                s SUFFIX:  Append all textual OIDs with SUFFIX before parsing
                S PREFIX:  Prepend all textual OIDs with PREFIX before parsing
-L LOGOPTS          Toggle various defaults controlling logging:
                e:          log to standard error
                o:          log to standard output
                n:          don't log at all
                f file:     log to the specified file
                s facility:  log to syslog (via the specified facility)

                (variants)
                [EON] pri:   log to standard error, output or /dev/null for level 'pri' and above
                [EON] p1-p2: log to standard error, output or /dev/null for levels 'p1' to 'p2'
                [FS] pri token:    log to file/syslog for level 'pri' and above
                [FS] p1-p2 token:  log to file/syslog for levels 'p1' to 'p2'
-C APPOPTS          Set various application specific behaviours:
                p:  print the number of variables found
                i:  include given OID in the search range
                I:  don't include the given OID, even if no results are returned
                c:  do not check returned OIDs are increasing
                t:  Display wall-clock time to complete the request
```

Figura 118 - Opções Net-SNMP

Anexo VI – Sintaxe comandos MySQL

SELECT

```
SELECT [STRAIGHT_JOIN]
    [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
    [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
[HIGH_PRIORITY]
    [DISTINCT | DISTINCTROW | ALL]
  expressão_select,...
  [INTO {OUTFILE | DUMPFILE} 'nome_arquivo' opções_exportação]
  [FROM tabelas_ref
    [WHERE definição_where]
    [GROUP BY {inteiro_sem_sinal | nome_col | formula} [ASC | DESC], ...
     [WITH ROLLUP]]
    [HAVING where_definition]
    [ORDER BY {inteiro_sem_sinal | nome_coluna | formula} [ASC | DESC], ...]
    [LIMIT [offset,] row_count | row_count OFFSET offset]
    [PROCEDURE nome_procedimento(lista_argumentos)]
    [FOR UPDATE | LOCK IN SHARE MODE]]
```

Figura 119 - Sintaxe comando SELECT

UPDATE

```
UPDATE [LOW_PRIORITY] [IGNORE] nome_tabela
  SET nome_coluna1=expr1 [, nome_coluna2=expr2 ...]
  [WHERE definição_where]
  [ORDER BY ...]
  [LIMIT row_count]
ou
UPDATE [LOW_PRIORITY] [IGNORE] nome_tabela [, nome_tabela ...]
  SET nome_coluna1=expr1 [, nome_coluna2=expr2 ...]
  [WHERE definição_where]
```

Figura 120 - Sintaxe comando UPDATE

DELETE

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM table_name
    [WHERE definição_where]
    [ORDER BY ...]
    [LIMIT row_count]

ou

DELETE [LOW_PRIORITY] [QUICK] [IGNORE] table_name[.*] [, table_name[.*] ...]
    FROM tabelas-referentes
    [WHERE definição_where]

ou

DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
    FROM nome_tabela[.*] [, nome_tabela[.*] ...]
    USING tabelas-referentes
    [WHERE definição_where]
```

Figura 121 - Sintaxe comando DELETE

CREATE DATABASE

```
CREATE DATABASE [IF NOT EXISTS] nome_bd
```

Figura 122 - Sintaxe comando CREATE

USE

```
USE nome_db
```

Figura 123 - Sintaxe comando USE

CREATE TABLE

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] nome_tabela [(definição_create,...)]
[table_options] [select_statement]

ou

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] nome_tabela [(]LIKE
nome_antigo_tabela[)];

definição_create:
  nome_coluna tipo [NOT NULL | NULL] [DEFAULT valor_padrão] [AUTO_INCREMENT]
        [[PRIMARY] KEY] [COMMENT 'string'] [definição_referência]
  | [CONSTRAINT [symbol]] PRIMARY KEY (index_col_name,...)
  | KEY [nome_indice] (index_nome_coluna,...)
  | INDEX [nome_indice] (index_nome_coluna,...)
  | [CONSTRAINT [symbol]] UNIQUE [INDEX] [index_name] (index_col_name,...)
  | FULLTEXT [INDEX] [nome_indice] (index_nome_coluna,...)
  | [CONSTRAINT [symbol]] FOREIGN KEY [index_name] (index_col_name,...)
        [definição_referência]
  |   CHECK (expr)

tipo:
      TINYINT[(tamanho)] [UNSIGNED] [ZEROFILL]
  |   SMALLINT[(tamanho)] [UNSIGNED] [ZEROFILL]
  |   MEDIUMINT[(tamanho)] [UNSIGNED] [ZEROFILL]
  |   INT[(tamanho)] [UNSIGNED] [ZEROFILL]
  |   INTEGER[(tamanho)] [UNSIGNED] [ZEROFILL]
  |   BIGINT[(tamanho)] [UNSIGNED] [ZEROFILL]
  |   REAL[(tamanho,decimais)] [UNSIGNED] [ZEROFILL]
  |   DOUBLE[(tamanho,decimais)] [UNSIGNED] [ZEROFILL]
  |   FLOAT[(tamanho,decimais)] [UNSIGNED] [ZEROFILL]
  |   DECIMAL(tamanho,decimais) [UNSIGNED] [ZEROFILL]
  |   NUMERIC(tamanho,decimais) [UNSIGNED] [ZEROFILL]
  |   CHAR(tamanho) [BINARY | ASCII | UNICODE]
  |   VARCHAR(tamanho) [BINARY]
  |   DATE
  |   TIME
  |   TIMESTAMP
  |   DATETIME
  |   TINYBLOB
  |   BLOB
  |   MEDIUMBLOB
  |   LONGBLOB
  |   TINYTEXT
  |   TEXT
  |   MEDIUMTEXT
  |   LONGTEXT
```

```
 |   ENUM(value1,value2,value3,...)
 |   SET(value1,value2,value3,...)

index_nome_coluna:
      nome_coluna [(tamanho)] [ASC | DESC]

definição_referência:
      REFERENCES nome_tabela [(index_nome_coluna,...)]
            [MATCH FULL | MATCH PARTIAL]
            [ON DELETE opção_referência]
            [ON UPDATE opção_referência]

opção_referência:
      RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT

opções_tabela: table_option [table_option] ...

opções_tabela:
  TYPE = {BDB | HEAP | ISAM | InnoDB | MERGE | MRG_MYISAM | MYISAM }
  | AUTO_INCREMENT = #
  | AVG_ROW_LENGTH = #
  | CHECKSUM = {0 | 1}
  | COMMENT = 'string'
  | MAX_ROWS = #
  | MIN_ROWS = #
  | PACK_KEYS = {0 | 1 | DEFAULT}
  | PASSWORD = 'string'
  | DELAY_KEY_WRITE = {0 | 1}
  | ROW_FORMAT = { DEFAULT | DYNAMIC | FIXED | COMPRESSED }
  | RAID_TYPE = { 1 | STRIPED | RAID0 } RAID_CHUNKS=#  RAID_CHUNKSIZE=#
  | UNION = (table_name,[table_name...])
  | INSERT_METHOD = { NO | FIRST | LAST }
  | DATA DIRECTORY = 'caminho absluto para o diretório'
  | INDEX DIRECTORY = 'caminho absluto para o diretório'
  | DEFAULT CHARACTER SET character_set_name [COLLATE collation_name]

instrução_select:
  [IGNORE | REPLACE] [AS] SELECT ...  (Alguma instrução válida)
```

Figura 124 - Sintaxe comando CREATE TABLE

LOAD DATA INFILE

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name.txt'
   [REPLACE | IGNORE]
   INTO TABLE nome_tabela
   [FIELDS
      [TERMINATED BY '\t']
      [[OPTIONALLY] ENCLOSED BY '']
      [ESCAPED BY '\\' ]
   ]
      [LINES
        [STARTING BY '']
        [TERMINATED BY '\n']
   ]
   [IGNORE número LINES]
   [(nome_coluna,...)]
```

Figura 125 - Sintaxe comando LOAD DATA INFILE

Anexo VII - Instalação PHP

Instalou-se a versão mais recente da página do PHP [PHP2008] pois, é necessário termos o ficheiro **configure** na pasta de instalação. Porém, previamente analisou-se se a compatibilidade entre a versão do servidor Apache e o PHP. Para a instalação executaram-se os seguintes comandos:

```
tar -zxvf php-5.2.1.tar.gz
cd php-5.2.1
./configure --prefix=/usr/local/php5 –with apxs2= /usr/local/apache2/bin/apxs --with-mysql=/usr/include/mysql --with-mysql
make
sudo make install
cp php.ini-dist /usr/local/lib/php.ini
```

Figura 126 - Comandos instalação PHP

Deve-se editar o ficheiro de configuração do servidor *Web* (Apache) **/usr/local/apache2/conf/httpd.conf** e adicionar à linha **DirectoryIndex index.html** os parâmetros **index.php default.php main.php**. Acrescenta-se também no final do ficheiro as linhas :

```
AddType application/x-httpd-php .php .phtml
AddType application/x-httpd-php-source .phps
```

Figura 127 - Comandos configuração Apache/PHP

Após alterações reinicializa-se o Apache. Pode-se fazer um teste de maneira a verificar se a instalação foi feita com sucesso:

- Edita-se um ficheiro, por exemplo com o nome **phpinfo.php**, na pasta **/usr/local/apache2/htdocs**, onde ficarão os códigos PHP e HTML do servidor.
- No ficheiro escrevem-se os seguintes comandos:

```
<?php
phpinfo();
?>
```

Figura 128 - Comandos teste

O código PHP começa sempre por **<?** e termina com **?>**.

- Salva-se o arquivo e acede-se o browser da seguinte forma: http://localhost/phpinfo.php

Na página criada é possível ver todas as informações relativas ao PHP e ao servidor Apache.

SInBAD

Estes anexos só estão disponíveis para consulta através do CD-ROM.
Queira por favor dirigir-se ao balcão de atendimento da Biblioteca.

Serviços de Documentação
Universidade de Aveiro